ANALISIS PERBANDINGAN KINERJA *FILE SYSTEM* GLUSTERFS DAN HDFS DENGAN SKENARIO DISTRIBUSI *STRIPED* DAN *REPLICATED*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
M. IRFAN SYAFI'I
NIM: 115090613111005

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

ANALISIS PERBANDINGAN KINERJA *FILE SYSTEM* GLUSTERFS DAN HDFS DENGAN SKENARIO DISTRIBUSI *STRIPED* DAN *REPLICATED*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
M. IRFAN SYAFI'I
NIM: 1150906013111005

Skripsi ini telah diuji dan dinyatakan lulus pada
03 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	*Dosen Pembimbing II*
Adhitya Bhawiyuga, S.Kom, M.Sc	Mahendra Data, S.Kom, M.Kom
NIK. 201405 890720 1 001	NIK. 201503 861117 1 001

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah SKRIPSI ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila ternyata didalam naskah SKRIPSI ini dapat dibuktikan terdapat unsur-unsur PLAGIASI, saya bersedia SKRIPSI ini digugurkan dan gelar akademik yang telah saya peroleh (SARJANA) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku. (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 03 Agustus 2018

M. Irfan Syafi'i

NIM: 115090600111023

KATA PENGANTAR

Puji syukur penulis ucapkan kehadirat Allah SWT yang telah melimpahkan segala Rahmat, Karunia, dan Hidayah-Nya sehingga penulis dapat menyelesaikan skripsi dengan judul "Analisa Perbandingan Kinerja *File System* GlusterFS dan HDFS dengan Distribusi *Striped* dan *Replicated*".

Skripsi ini diajukan sebagai salah satu syarat memperoleh gelar Sarjana Komputer di Fakultas Ilmu Komputer, Universitas Brawijaya Malang. Atas terselesaikannya skripsi ini, penulis mengucapkan terima kasih kepada beberapa pihak di antaranya:

1. Bapak Adhitya Bhawiyuga, S.Kom., M.Sc dan Mahendra Data, S.Kom., M.Kom. selaku dosen pembimbing I dan pembimbing II yang telah banyak memberikan ilmu, bantuan, bimbingan, dan motivasi dalam penyelesaian skripsi ini.
2. Arif Nudin dan Binti Khoirun Nikmah, selaku kedua orang tua penulis yang selalu memberikan doa, motivasi, dukungan moril dan materil sebagai penyemangat dalam menyelesaikan skripsi ini.
3. Bapak Nanang Yudi Setiawan S.T, M.Kom. selaku Dosen Pembimbing Akademik yang selalu membimbing kegiatan perkuliahan dari awal masuk sampai lulus kuliah.
4. Bapak Tri Astoto Kurniawan, S.T., M.T., Ph.D selaku Ketua Jurusan Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
5. Bapak Prasetyo Iskandar, S.T dan Ibu Wiwin Lukitohadi, S.H, S.Psi, CHRM selaku staff Bimbingan Konseling yang selalu menyemangati dan memotivasi penulis dalam pengerjaan skripsi.
6. Segenap Bapak dan Ibu dosen yang telah mendidik dan mengajarkan ilmunya kepada penulis selama menempuh pendidikan di Fakultas Ilmu Komputer Universitas Brawijaya Malang.

7. Teman-teman Informatika angkatan 2011 yang selalu mendukung dan berbagi ilmu dari awal perkuliahan sampai tahap akhir penyelesaian skripsi.
8. Segenap staf dan karyawan di Fakultas Ilmu Komputer yang telah banyak membantu penulis dalam pelaksanaan penyusunan skripsi ini.
9. Semua pihak yang telah membantu terselesaikannya skripsi ini yang tidak dapat penulis sebutkan satu per satu.

Penulis menyadari bahwa skripsi ini masih memiliki kekurangan dan jauh dari sempurna, karena keterbatasan materi dan pengetahuan yang dimiliki penulis. Maka, saran dan kritik yang membangun dari semua pihak sangat diharapkan demi penyempurnaan selanjutnya. Semoga skripsi ini dapat bermanfaat dan berguna bagi semua pihak, baik penulis maupun pembaca, dan semoga Allah SWT meridhoi dan dicatat sebagai amalan ibadah, Amin.

Malang, 03 Agustus 2018

M. Irfan Syafi'i
senpai.irfan@gmail.com

ABSTRAK

Big Data merupakan istilah yang digunakan untuk menggambarkan pertumbuhan data yang besar, baik data terstruktur maupun data tidak terstrukur. *Big Data* mempuyai tiga karakeristik utama yaitu *volume, velocity,* dan *variety*. Permasalahan yang timbul dengan semakin berkembangnya Big Data adalah bagaimana cara menyimpan data tersebut. Data yang terus tumbuh membesar setiap waktu membutuhkan ruang penyimpanan yang besar pula. Hal ini tentu tidak akan mampu bila ruang penyimpanan tersebut berada dalam satu mesin *(single node/host)*. Sistem file terdistribusi merupakan modul penyimpanan dan pengelolaan file yang terdiri dari banyak mesin *(multi node/host)*. Penelitian ini bertujuan untuk membandingkan kinerja dua *file system* yakni GlusterFS dan HDFS dalam melakukan penyimpanan file terdistribusi dengan skenario distribusi *striped* dan *replicated*. Penelitian berbatas pada pengukuran kinerja *file system* dalam melakukan operasional *write/read file*. Hasil pengujian menunjukkan bahwa GlusterFS memiliki kinerja yang lebih ringan dalam melakukan operasional *write file* dengan perolehan *throughput* sebesar 44,54 MBps, waktu eksekusi selama 58,54 detik, beban penggunaan CPU sebesar 54,83% dan penggunaan memori sebesar 3,6%. Sedangkan HDFS, memiliki kinerja optimal saat operasional *read file* diperoleh hasil rata-rata *throughput* sebesar 194,37 MBps, waktu eksekusi selama 16,01 detik, beban penggunaan CPU sebesar 86,9% dan penggunaan memori sebesar 18,5%.

Kata kunci: *Big Data, Sistem File Terdistribusi, GlusterFS, HDFS*

ABSTRACT

Big Data is a term used to describe the growth of large data, both structured data and data not tersrukur. Big Data has three main characteristics: volume, velocity, and variety. The problem that arises with the development of Big Data is how to store the data. Data that continues to grow enlarged each time requires a large storage space as well. This certainly will not be able if the storage space is in one machine (single node / host). Distributed file system is a storage and file management module consisting of multiple machines (multi node / host). This study aims to compare the performance of two file systems, GlusterFS and HDFS in distributed file storage with striped and replicated distribution scenarios. The study is limited to the measurement of file system performance in performing write / read file operational. The test results show that the performance GlusterFS have a lighter in performing write file operations with the acquisition of 44.54 MBps throughput, the execution time for 58.54 seconds, CPU usage of 54.83% and memory usage of 3.6%. HDFS has the optimal performance on operational write files, obtained the average of throughput for 194.37 MBps, execution time for 16.01 seconds, CPU usage of 86.9% and memory usage of 18.5%.

Keywords: *Big Data, Distributed File System, GlusterFS, HDFS*

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1 PENDAHULUAN

1.1 Latar belakang

Big Data merupakan istilah yang digunakan untuk menggambarkan pertumbuhan data yang besar, baik data terstruktur (numerik, tanggal, waktu) maupun data tidak terstrukur (video, audio, dokumen). Istilah *Big Data* kali pertama pernah dipublikasikan oleh Fremont Rider, seorang pustakawan Amerika dari Westleyan University, pada tahun 1944. Dia memperkirakan bahwa *volume* koleksi universitas di Amerika akan mencapai 200 juta kopi di tahun 2040 (Narendra, 2017). Pada tahun 2000-an, fenomena *Big Data* mulai dikenal dan diteliti oleh berbagai kalangan, ketika seorang analisis industri, Doug Laney, menyampaikan konsep *Big Data* yang terdiri dari tiga bagian penting, yaitu *volume*, velocity, dan variety (Maryanto, 2017). *Volume* (ukuran), mengacu pada ukuran data yang semakin bertambah dari waktu ke waktu. *Velocity* (kecepatan), kecepatan data yang mengalir dari waktu ke waktu juga terus meningkat dan harus ditangani tepat waktu. *Variety* (beragam), kumpulan data yang terus bertambah besar memiliki format yang berbeda-beda (data yang terstruktur dan tidak terstruktur).

Permasalahan yang timbul dengan semakin berkembangnya *Big Data* adalah bagaimana cara penyimpanannnya. Data yang terus tumbuh membesar setiap waktu membutuhkan ruang penyimpanan yang besar pula. Hal ini tentu tidak akan mampu bila ruang penyimpanan tersebut berada dalam satu mesin (*single node/host*). Kecepatan data yang mengalir setiap saat yang juga membutuhkan penangan cepat serta beragamnya data yang terus berkumpul membuat sebuah ruang penyimpanan tidak mampu lagi ditingkatkan. Ruang penyimpanan yang dibutuhkan untuk menangani *Big Data* harus terdiri dari banyak mesin yang saling terhubung satu sama lain dan mampu untuk terus ditingkatkan kinerjanya mengikuti pertumbuhan data saat ini. Solusi alternatif untuk mengatasi persoalan tersebut adalah dengan membangun modul sistem *file* terdistribusi. Sistem file terdistribusi merupakan modul penyimpanan dan pengelolaan file yang terdiri dari banyak mesin (*multi node/host).* Sehingga apabila terdapat sebuah data berukuran besar masuk, data tersebut akan dipecah menjadi *block* kecil dan disimpan (didistribusikan) ke dalam node yang terdapat dalam sistem file. Beberapa perusahaan *platform* telah mengembangkan sistem file terdistribusi untuk mengelola Big Data. Beberapa sistem file terdistribusi yang ada saat ini di antaranya seperti Ceph, GFS, LizardFS, *Windows Distributed File system,* GlusterFS, dan HDFS. Dari beberapa sistem file terdistribusi tersebut, 2 di antaranya memiliki arsitektur dan performa kinerja yang hampir sama, yakni HDFS (*Hadoop Distributed File System*) yang dikembangkan oleh *Apache Software Foundation* dan GlusterFS (*Gluster File System*) yang dikembangkan oleh RedHat.Inc.

HDFS (*Hadoop Distributed File System*) merupakan sistem file terdistribusi yang dikembangkan oleh Apache Software Foundation. HDFS merupakan suatu *framework* yang memungkinkan pengolahan data berukuran besar secara

terdistribusi dengan melibatkan banyak klaster komputer. HDFS mampu membagi proses komputasi dengan para klasternya tanpa tumpang tindih dan secara cepat. Proses komputasi tersebut melibatkan *MapReduce* dan *JobTracker* yang hanya dimiliki dan merupakan komponen penyusun HDFS. Dengan membagi beban kerja antar klaster komputer, maka tidak perlu khawatir dengan rusak atau hilangnya data karena adanya komputer yang mati.

GlusterFS (*Gluster File System*) adalah sebuah apilkasi *open source* yang dikembangkan oleh RedHat.inc. GlusterFS merupakan *clustered file system* yang dapat beroperasi dengan kapasitas *petabyte* dan mampu menangani ribuan *client.* GlusterFS memiliki 2 komponen utama yaitu *gluster server* dan *gluster client. Gluster server* merupakan ruang penyimpanan utama yang di dalamnya terdapat satu jaringan *cluster.* Jaringan *cluster* tersebut menggabungkan kapasitas ruang penyimpanan beberapa *node server* menjadi penyimpanan tunggal yang disebut *volume. Gluster client* berfungsi untuk mengakses ruang penyimpanan utama tersebut. *Gluster client* juga mampu melakukan penambahan atau perubahan file pada ruang penyimpanan utama (Sulistyo, 2014).

Dari uraian di atas dapat diketahui bahwa HDFS dan GlusterFS merupakan sistem file terdistribusi yang bisa dijadikan sebagai solusi alternatif untuk pengelolaan *Big Data.* Hal inilah yang memunculkan pemikiran untuk dilakukannya analisis perbandingan kinerja antara GlusterFS dan HDFS untuk mengetahui kemampuan masing-masing *file system* dengan melakukan penulisan dan pembacaan suatu file yang disimpan secara paralel antar komputer. Analisis perbandingan akan diukur melalui parameter perbandingan *throughput,* eksekusi waktu*,* beban penggunaan CPU dan *memory usage*. Hasil yang didapat melalui analisis perbandingan ini diharapkan mampu digunakan sebagai acuan oleh perusahaan atau instansi dalam pengelolaan *Big Data.*

1.2 Rumusan masalah

Dari uraian latar belakang di atas maka dapat disusun rumusan masalah dalam penelitian ini, sebagai berikut:

1. Bagaimana perancangan lingkungan pengujian perbandingan kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *striped* dan *replicated*?
2. Apa saja parameter yang digunakan untuk pengukuran perbandingan kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *striped* dan *replicated*?
3. Bagaimana skenario pengujian yang digunakan dalam mengukur perbandingan kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *striped* dan *replicated*?
4. Bagaimana analisis hasil dari pengujian perbandingan kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *striped* dan *replicated*?

1.3 Tujuan

Tujuan yang ingin dicapai dalam penelitian ini adalah sebagai berikut:

1. Membangun lingkungan pengujian untuk membandingkan kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *striped* dan *replicate.*
2. Mengetahui perbedaan kinerja *file system* GlusterFS dan HDFS dalam melakukan operasional *write/read file* dengan membandingkan 4 parameter pengukuran kinerja, yaitu *throughput,* waktu eksekusi*,* beban penggunaan CPU dan *memori usage*.
3. Melakukan pengukuran kinerja operasional *write/read file* dengan menggunakan 5 skenario pengujian yang berbeda, yakni variasi ukuran file, variasi jumlah node aktif, variasi ukuran block, variasi jumlah replikasi dan penangan *fault system*.
4. Melakukan analisis terhadap hasil pengujian untuk membandingkan kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *striped* dan *replicated*.
5. Mengambil kesimpulan *file system* mana yang lebih bagus untuk digunakan dalam penyimpanan file terdistribusi, serta mengetahui faktor apa saja yang mempengaruhi kinerja dari masing-masing *file system*.

1.4 Manfaat

Hasil dari penelitian ini diharapkan dapat memberikan manfaat pada administrator *server* dan pengguna dalam penyimpanan file terdistribusi untuk solusi alternatif pengelolaan *Big Data*. Dengan mengetahui perbedaan dan kemampuan dari GlusterFS dan HDSF tentunya pengguna dapat mengolah, menyimpan serta berbagi berkas dan sumber daya *Big Data* dengan lebih efisien.

1.5 Batasan masalah

Ruang lingkup pembahasan dalam penelitian ini meliputi hal-hal berikut ini:

1. Penelitian yang dilakukan berupa simulasi penyimpanan file terdistribusi dengan besar ukuran file tidak lebih dari 5GB.
2. Penelitian ini fokus pada pembahasan membandingkan kinerja dari GlusterFS dan HDFS dalam melakukan penyimpanan terdistribusi dengan skenario distribusi *striped* dan *replicated*.
3. Pengujian dilakukan di ruangan laboratorium multimedia gedung E lantai 2, Fakultas Ilmu Komputer, Universitas Brawijaya dengan menggunakan masing-masing 3 host virtual yang terdiri dari 1 node *master* dan 2 node *slave*.

1.6 Sistematika Pembahasan

Untuk mencapai tujuan yang diharapkan, maka sistematika penulisan yang disusun dalam skripsi ini adalah sebagai berikut:

BAB 1 Pendahuluan

Bab ini berisi tentang latar belakang, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah dan sistematika penulisan dari penelitian "Analisis Perbandingan Kinerja File System GlusterFS dan HDFS Skenario Distribusi Striped dan Replicated".

BAB 2 Landasan Kepustakaan

Bab ini berisi uraian dan pembahasan tentang teori, konsep, model, metode atau sistem dari literatur ilmiah, yang berkaitan dengan penelitian "Analisis Perbandingan Kinerja *File System* GlusterFS dan HDFS Skenario Distribusi *Striped* dan *Replicated*". Bab ini dibagi menjadi dua yaitu kajian pustaka dan dasar teori. Kajian pustaka mengkaji perbandingan antara penelitian yang diusulkan dengan penelitian yang sudah ada sebelumnya atau dari beberapa sumber resmi yang dapat digunakan sebagai literatur. Dasar teori membahas teori-teori yang mendukung dalam analisis perbandingan kinerja GlusterFS dan HDFS sebagai metode pengelolaan file terdistribusi.

BAB 3 Metodologi

Bab ini membahas tentang tahap-tahap kinerja yang digunakan dalam penelitian ini. Tahap-tahap kinerja tersebut antara lain studi literatur, analisis kebutuhan, perancangan lingkungan pengujian, pengujian serta analisis hasil dari perbandingan *kinerja file system* GlusterFS dan HDFS dalam penyimpanan file terdistribusi.

BAB 4 Perancangan Lingkungan Pengujian

Bab ini membahas tentang perancangan lingkungan pengujian. Lingkungan pengujian ini disiapkan guna melakukan pengujian perbandingan kinerja *file system* GlusterFS dan HDFS.

BAB 5 Pengujian dan Analisis Hasil

Bab ini membahas tentang pengujian dan analisis hasil yang telah dilakukan dengan membandingkan kinerja GlusterFS dan HDFS. Hasil yang didapat kemudian akan dianalisis dan kemudian akan ditarik kesimpulan, *file system* mana yang memiliki kinerja paling baik dalam melakukan pengelolaan file terdistribusi.

Bab 6 Penutup

Bab ini membahas kesimpulan dari pembahasan rumusan masalah. Kesimpulan yang diambil merupakan hasil dari keseluruhan analisis perbandingan kinerja *file system* GlusterFS dan HDFS yang telah dilakukan dalam penelitian ini. Selain itu, disertakan pula saran yang ditujukan untuk penelitian selanjutnya.

BAB 2 KAJIAN PUSTAKA DAN LANDASAN TEORI

Pada bab dua, terdiri dari kajian pustaka dan dasar teori. Kajian pustaka merupakan pembahasan penelitian yang telah ada dan diusulkan. Dasar teori membahas teori apa saja yang diperlukan untuk menyusun penelitian yang diusulkan.

2.1 Kajian Pustaka

Penelitian sebelumnya yang pernah dilakukan oleh Khusumanegara (2011) yang berjudul, "Analisis Performa Kecepatan MapReduce pada Hadoop Menggunakan TCP *Packet Flow Analysis".* Penelitian tersebut bertujuan untuk mengetahui apa yang mempengaruhi kecepatan distribusi file dalam Hadoop dengan enam skenario berdasarkan topologi yang dirancang. Hasilnya bahwa kecepatan distribusi file dalam Hadoop dipengaruhi oleh penambahan *physical machine*. Pada ukuran file 512 MB, 1 GB, 1.5 GB, dan 2 GB, penambahan physical machine dapat mempercepat kecepatan rata-rata MapReduce pada masing-masing ukuran file sebesar 161.34, 328.00, 460.20 dan 525.80 detik. Sedangkan penambahan *virtual machine* justru memperlambat distribusi file. Pada ukuran file 512 MB, 1 GB, 1.5 GB, dan 2 GB, penambahan virtual machine dapat memperlambat kecepatan rata-rata MapReduce pada masing-masing ukuran file sebesar 164.00, 504.34, 781.27 dan 1070.46 detik

Penelitian lain yang dilakukan oleh Donvito, et al. (2014) yang berjudul, "*Testing of several distributed file-systems (HDFS, Ceph and GlusterFS) for supporting the HEP experiments analysis".* Penelitian tersebut dilakukan untuk menguji beberapa *file system* terdistribusi untuk mendukung analisis pengujian HEP. Dalam penelitian ini terlihat perbedaan ketiga *file system* terdistribusi yang digunakan. Perbedaan tersebut ditunjukkan pada Tabel 2.1 sebagai berikut:

Tabel 2.1: Hasil uji kecepatan distribusi file dengan iozone (MB/s)

	HDFS	Ceph	GlusterFS
Initial Write	239.72	51.06	306.34
Re-Write	ERROR	60.05	406.90
Random Write	ERROR	7.00	406.33
Read	155.18	101.58	688.06
Re-Read	151.33	133.61	711.46
Random Read	29.06	12.05	284.00

Sumber: (Donvito, 2014)

2.2 Landasan Teori

2.2.1 Big Data

Big Data adalah suatu trend/isu yang mencakup area luas dalam dunia bisnis dan teknologi. *Big Data* bukanlah sebuah teknologi, namun merujuk pada perkembangan teknologi yang melibatkan beragam bentuk data, cepat berubah, atau berukuran super besar sehingga dinilai terlalu sulit bagi teknologi, kemampuan, maupun infrastruktur setandar umum untuk dapat menanganinya secara efektif (Megantara & Warnars, 2016). Dengan kata lain, *Big Data* memiliki 3 karakteristik utama, yakni ukuran (*volume*), kecepatan (*velocity*), atau ragam (*variety*).

Ukuran *(volume)*, karakteristik *Big Data* yang pertama dapat digambarkan dengan uraian berikut ini:

- Pada tahun 2000 lalu, PC biasa pada umumnya memiliki kapasitas penyimpanan sekitar 10 gigabytes.
- Saat ini jaringan media sosial, contohnya Facebook, menyedot sekitar 500 terabytes data baru setiap harinya.
- Sebuah pesawat Boeing 737 menghasilkan sekitar 240 terabytes data penerbangan dalam satu penerbangan lintas negara.

Jadi bisa dikatakan bahwa ukuran *big Data* akan terus bertambah setiap waktunya, seiring dengan semakin canggih dan majunya teknologi.

Kecepatan *(velocity)*, karakteristik *Big Data* yang kedua mengacu pada kecepatan lalu lintas data setiap waktunya. Misalnya, banyaknya pengguna internet setiap waktunya dalam melakukakan aktifitas *surfing* di dunia maya, sistem game *online* dapat melayani jutaan pengguna secara bersamaan, yang masing-masing memberikan sejumlah input per detiknya, peralatan sensor dan perangkat-perangkat pada infrastruktur menghasilkan log data secara *real time* dan lain sebagainya.

Ragam *(variety)*, karakteristik *Big Data* yang ketiga adalah beragamnya bentuk data yang terkumpul. *Big Data* tidak hanya menyangkut data jenis angka, tanggal ataupun rangkaian teks. *Big Data* juga meliputi data-data ruang / geospatial, data 3D, audio / video, data-data teks terstruktur maupun tak berstruktur *(log file)* dan media sosial.

2.2.2 Distributed File System

Distributed File System (Sistem File Terdistribusi) adalah *file system* yang mendukung *sharing files* dan sumber daya dalam bentuk penyimpanan berkesinambungan (berkala) di dalam suatu modul jaringan (Sekarwati, 2005). Di dalam sistem file terdistribusi, suatu file disimpan secara eksplisit dan akan tetap ada sampai ada perintah proses penghapusan. Dengan demikian suatu file tersebut dapat digunakan sebagai penanda bahwa sistem file file tersebut masih terhubung.

Sistem file terdistribusi melayani pemrosesan data secara tidak langsung, sehingga memungkinkan pengguna untuk menyimpan dan berbagi data. Sistem file terdistribusi merupakan sebuah mekanisme sistem yang juga melibatkan kinerja dari sisi *server* dan *client*, di mana data akan disimpan di beberapa komputer server yang disebut *node.* Penyimpanan data secara distribusi (disebar) di beberapa *node* disebut dengan replikasi. *Server* akan berperan sebagai penyedia layanan (*file service*) dalam proses distribusi file, sedangkan di sisi *client* terdapat *interface* untuk *file service* seperti membuat file (*create*), menghapus (*delete*) dan membaca (*read*) file. Komponen *hardware* yang digunakan sebagai pengontrolnya adalah sebuah *local storage* (*disk drive*/HDD) dan juga menjadi tempat penyimpanan file-file serta tempat untuk *client* meminta *retrieve file* (Scribd, 2013).

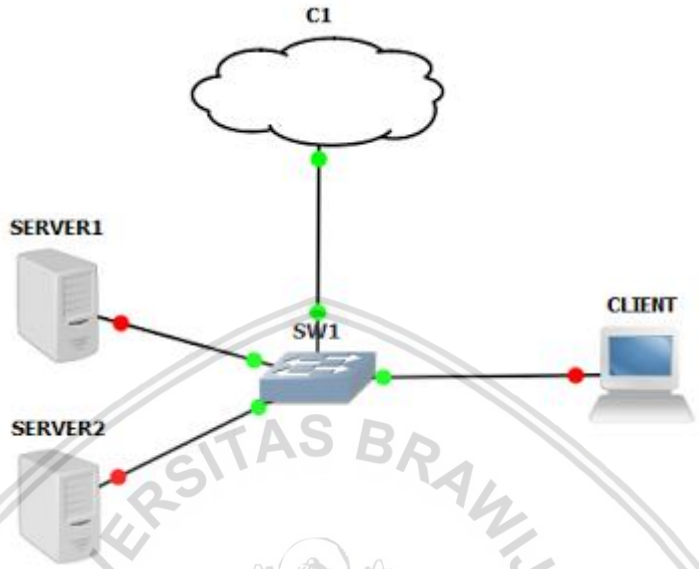
Secara ideal, sistem file terdistribusi terlihat sebagai bentuk sistem file terpusat dengan adanya faktor keseragaman (*multiplicity*) dan penyebaran server serta perangkat penyimpanan yang tidak terlihat oleh *user*. Program antarmuka yang dijalankan di sisi *client* tidak akan membedakan antara file lokal dan *remote*. Semuanya tergantung dari implementator *file system* untuk mengalokasikan file-file dan menyusun transportasi data.

Sistem file terdistribusi mendukung pengiriman informasi dalam bentuk file yang disimpan di dalam sebuah *mount-point storage* melalui jaringan intranet. *File service* yang telah dirancang dengan baik menyediakan akses ke file yang disimpan pada server dengan kinerja yang Sama atau bahkan lebih baik dari file yang disimpan pada local disk. Desainnya disesuaikan dengan performance dari jaringan lokal dan oleh karena itulah menjadi yang paling efektif dalam menyediakan pengiriman storage untuk digunakan di dalam *intranet* (Scribd, 2013).

2.2.3 GlusterFS

GlusterFS (*Gluster File System*) adalah sebuah aplikasi *open source* yang digunakan untuk memanajemen sistem berkas terdistribusi (*clustered file-system*) dan mampu diskalakan sampai dengan beberapa *peta-bytes*. GlusterFS menggabungkan disk, memori dan pengolahan data dari beberapa modul server ke dalam sebuah ruang penyimpanan tunggal (Sulistyo, & Prasetyo, 2014). Arsitektur GlusterFS bersifat modular yang memungkinkan seorang administrator menambah atau mengurangi modul server sesuai dengan kebutuhan pengguna.

GlusterFS merupakan *file system* yang dirancang untuk *cloud computing* dengan performa tinggi. Gambar 2.1 berikut ini memperlihatkan topologi dari GlusterFS.



Gambar 2.1: Topologi umum GlusterFS

Sumber: (Gluster, 2011)

Keterangan:

- C1: *cloud,* yang dimaksud *cloud* di sini adalah *volume* yang digunakan untuk menyimpan file. *Volume* tersebut terdapat pada *node server*
- SW1: *switch,* yang berperan sebagai *switch* (penghubung) adalah direktori *mount point* yang berada pada *node client*

Dalam GlusterFS ruang ruang penyimpanan data disebut *volume*. GlusterFS memiliki kemampuan mengatur kuota penggunaan ruang disk dengan direktori atau *volume*. Admin dapat mengendalikan pemanfaatan ruang disk pada tingkat direktori dan atau *volume* dengan menetapkan batas-batas untuk ruang disk yang dialokasikan di setiap tingkat dalam *volume* dan hierarki direktori. GlusterFS memiliki dua konsep dasar yakni:

1. ***GlusterFS Distributed***, di mana saat *client* membuat beberapa file, file tersebut akan terbagi menjadi 2. Sebagian file akan disimpan di server 1 dan yang lain di server 2.
2. ***GlusterFS Replicated***, di mana ketika *client* membuat beberapa file, file tersebut akan tersimpan dan terbaca oleh server 1 maupun server 2.

Konsep dan terminologi GlusterFS yang sudah dijelaskan dalam buku pedoman resminya (Gluster, 2011) adalah sebagai berikut :

a) ***Brick***

Merupakan sebuah direktori yang di *shared* (penghubung) di antara *trusted storage pool* (antar sesama *node server* dan antar *node server-client* GlusterFS).

b) ***Trusted Storage Pool***

Sebutan untuk kumpulan *file shared*/suatu direktori yang didasarkan pada desain protokol (pembuatan *volume*).

c) ***Block Storage***: *Device* yang dilewati data yang bergerak melintasi sistem dalam bentuk *block*.

d) ***Cluster***

Dalam istilah manajemen storage, *cluster* dan *trusted storage pool* mempunyai makna yang sama dari kolaborasi storage server berdasarkan protokol yang didefinisikan.

e) ***Fuse***

Sebuah modul kernel yang memungkinkan *user* untuk dapat membuat *file system* diatas kernel tanpa melibatkan dan mengganggu proses berjalannya *kernel code*.

f) ***Glusterd***

Sebuah daemon untuk mengontrol dan memanajemen GlusterFS serta merupakan *backbone* dari GlusterFS yang akan berjalan sepanjang waktu selama server dalam keadaan aktif.

g) **POSIX**

Portable Operating System Interfaces adalah sebuah standard yang didefinisikan oleh IEEE sebagai solusi dari kompatibilitas antara *unix-varian* dalam bentuk API.

h) ***Subvolume***

Merupakan sebuah *brick* yang telah diproses setelah ditranslasikan.

i) ***Translator***

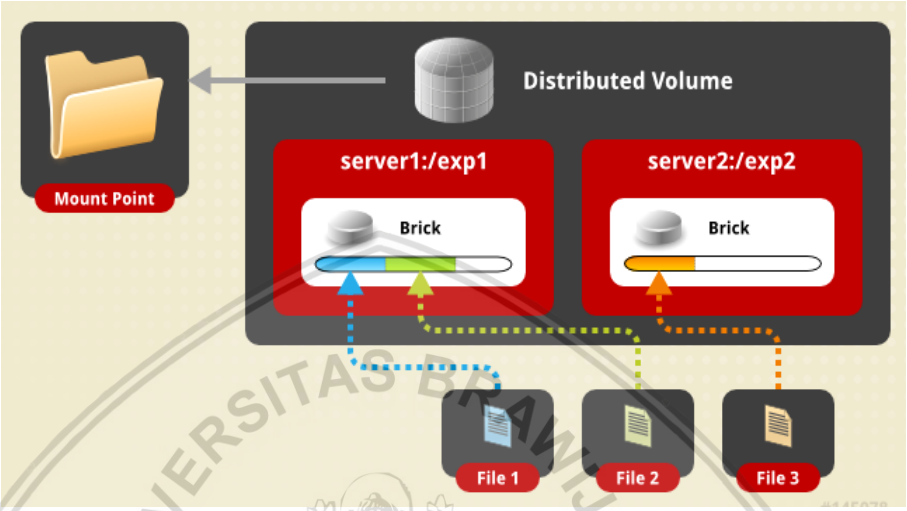
Merupakan sekumpulan kode yang menjalankan *basic action* yang dipicu oleh *user* dari *mount point*. Translator dapat menghubungkan satu atau lebih *subvolume*.

j) ***Volume***

Sebuah kumpulan *logical storage* dari *brick*. Semua proses operasi dalam *volume* adalah berdasarkan tipe (*distribute, replicate, stripe*) yang didefinisikan oleh *user*.

GlusterFS dapat didesain dalam beberapa mode. Mode yang umum dipakai adalah sebagai berikut ini (Sulistyo, 2014):

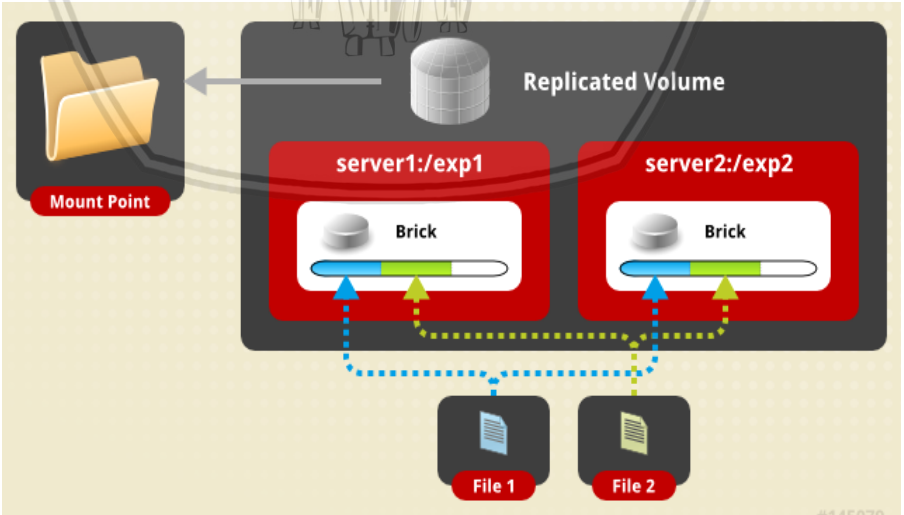
1. *Distributed*: *Volume* distribusi, di mana pengguna perlu menentukan skala penyimpanan karena dalam *volume* distribusi file akan disebar secara acak. Mekanisme mode *distributed* seperti yang diperlihatkan Gambar 2.2 berikut:



Gambar 2.2 : GlusterFS Distributed

Sumber: docs.gluster.org

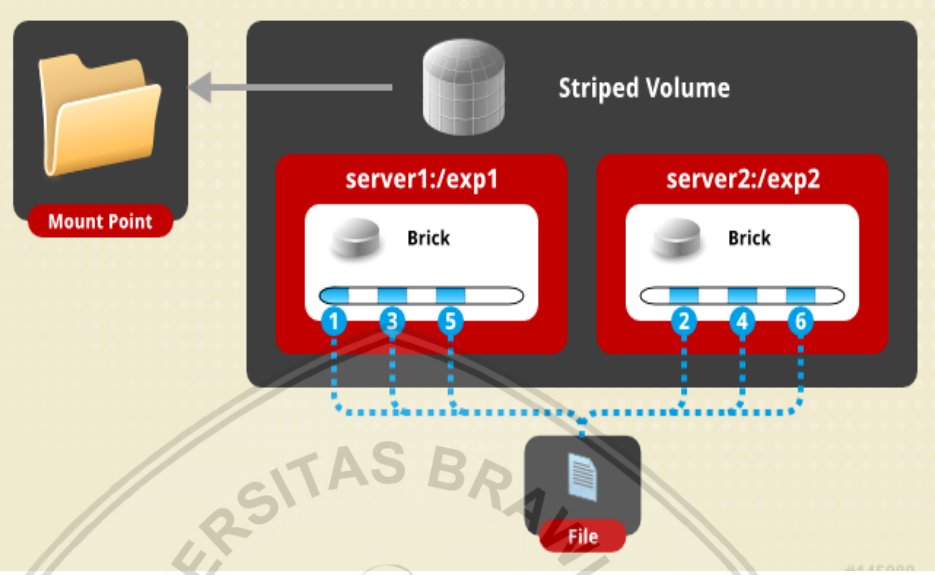
2. *Replicated*: *Volume* direplikasi, artinya file yang disimpan akan dibuat salinan yang serupa dengan file asli di beberapa penyusunan *volume*. Mekanisme distribusi *replicated* seperti yang diperlihatkan Gambar 2.2 berikut:



Gambar 2.3 : GlusterFS Replicated

Sumber: docs.gluster.org

3. *Striped*: Memecah file di antara simpul (*node*) dalam *cluster*. Biasanya metode *striped* digunakan untuk mengakses file yang sangat besar. Mekanisme distribusi *striped* seperti yang diperlihatkan Gambar 2.4 berikut:



Gambar 2.4 : GlusterFS Striped

Sumber: docs.gluster.org

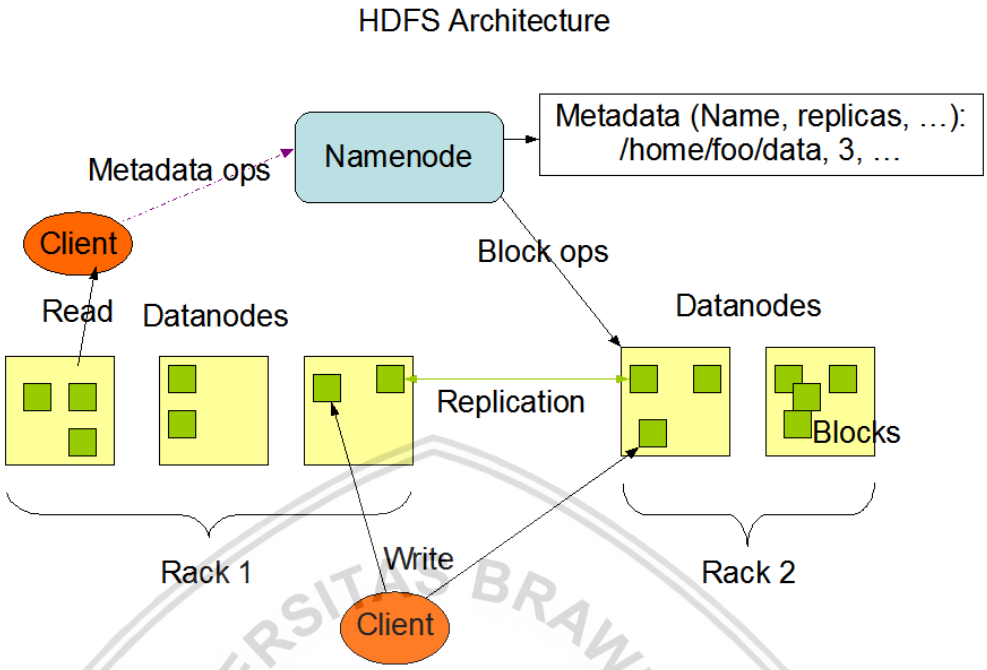
2.2.4 HDFS

HDFS (*Hadoop Distributed File System*) adalah *file system* berbasis Java yang terdistribusi langsung pada Hadoop. Sebagai *file system* terdistribusi HDFS digunakan untuk mengolah data dalam ukuran besar karena memiliki ukuran *block size* yang lebih besar dibanding *file system* lainnya. HDFS berguna untuk menangani data dalam jumlah besar yang disimpan dan tersebar di dalam banyak komputer yang berhubungan yang biasa disebut dengan *cluster*. *File system* terdistribusi pada Hadoop dapat diartikan sebagai *file system* yang menyimpan data tidak dalam satu *Hard Disk Drive* (HDD) atau media penyimpanan lainnya, tetapi data dipecah-pecah (*file* dipecah dalam bentuk *block* dengan ukuran 64 MB – bisa dikonfigurasi besarnya) dan disimpan tersebar dalam suatu *cluster* yang terdiri dari beberapa komputer.

2.2.4.1 Model Data HDSF

HDFS menyimpan data dengan cara membelahnya menjadi potongan-potongan data yang berukuran *default* 64MB atau 128MB. Potongan-potongan data tersebut disebut dengan istilah *block*. Meskipun data yang ada disimpan secara tersebar ke beberapa *node*, namun dari kacamata *user*, data tersebut tetap terlihat seperti halnya kita mengakses *file* pada satu komputer. *File* yang secara fisik tersebar dalam banyak komputer dapat diperlakukan layaknya memperlakukan *file* dalam satu komputer (Khusumanegara, 2014).

2.2.4.2 Komponen HDFS



Gambar 2.5 : Komponen HDFS

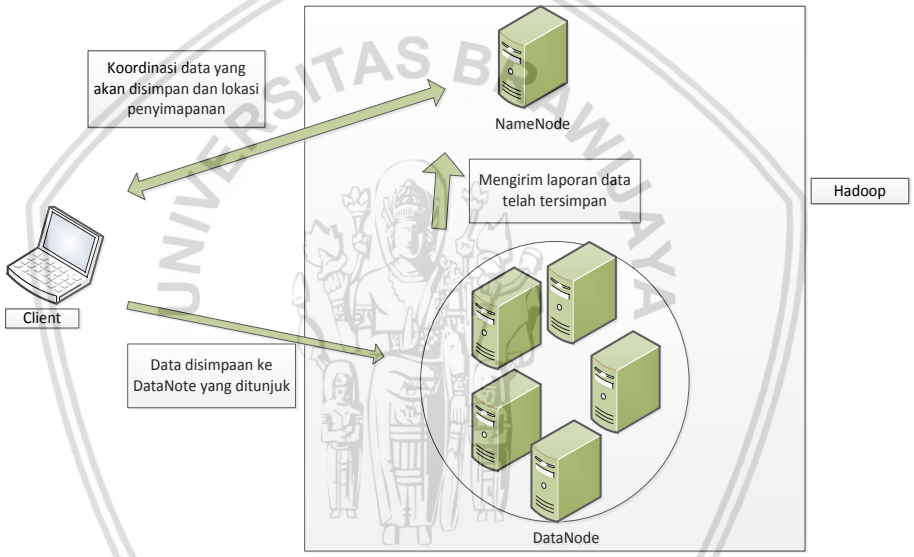
Sumber: hadoop.apache.org

Gambar 2.5 di atas menggambarkan komponen-komponen penyusun HDFS. Sebagai distributed *file system*, HDFS memiliki komponen-komponen utama berupa *NameNode* dan *DataNode*. *NameNode* adalah sebuah komputer yang bertindak sebagai master, sedangkan *DataNode* adalah komputer-komputer dalam *Hadoop Cluster* yang bertugas sebagai *slaves* atau anak buah. *NameNode* bertanggungjawab menyimpan informasi tentang penempatan block-block data dalam *Hadoop Cluster*. Ia bertanggungjawab mengorganisir dan mengontrol block-block data yang disimpan tersebar dalam komputer-komputer yang menyusun *Hadoop Cluster*. Sedangkan *DataNode* bertugas menyimpan block-block data yang dialamatkan kepadanya, dan secara berkala melaporkan kondisinya kepada *NameNode*. Laporan berkala *DataNode* kepada *NameNode* ini disebut *Heartbeat*. Berdasarkan *Heartbeat* ini *NameNode* dapat mengetahui dan menguasai kondisi cluster secara keseluruhan. Sebagai balasan atas *Heartbeat* dari *DataNode*, *NameNode* akan mengirimkan perintah kepada *DataNode*. Jadi, dalam HDFS, *NameNode* adalah boss yang mengatur dan mengendalikan atau me-manage *Hadoop Cluster*. Sedangkan, *DataNode* adalah pekerja atau karyawan yang bertugas menyimpan data dan melaksanakan perintah dari *NameNode* (Yulianto, 2015).

Secondary NameNode adalah daemon yang berfungsi melakukan monitoring keadaan dari cluster HDFS. Sama seperti *NameNode*, pada setiap cluster yang ada terdapat satu *Secondary NameNode*, yang berada pada master node. *Secondary NameNode* ini juga berfungsi untuk membantu dalam meminimalkan *down time* dan hilangnya data yang terjadi pada HDFS.

2.2.4.3 Prosedur Menyimpan dan Membaca Data dalam HDFS

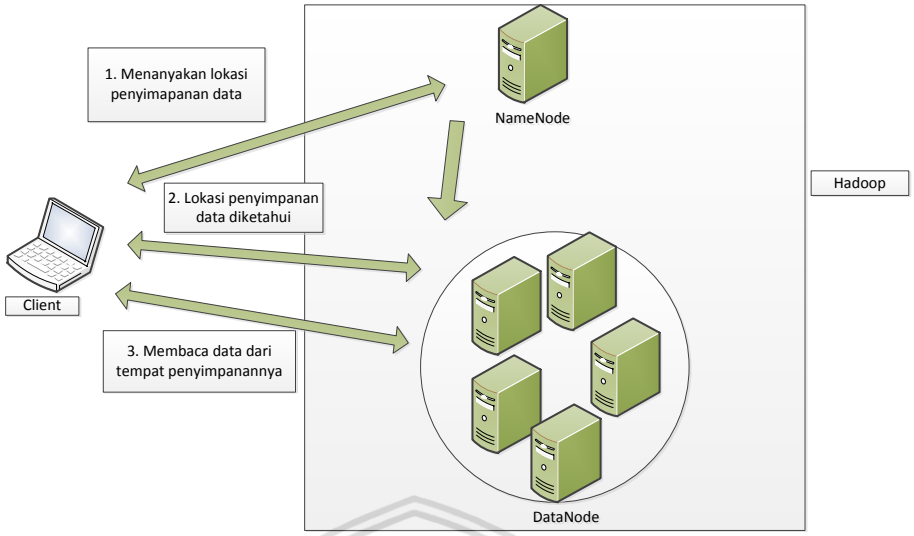
Untuk menyimpan data dengan menggunakan HDFS, diperlukan sebuah host *client* yang sudah terhubung dengan *Hadoop Cluster*. Pada saat dilakukan eksekusi perintah penyimpanan, komputer *client* akan berkomunikasi dengan *NameNode* untuk memberi tahu bahwa ada file yang hendak disimpan ke dalam HDFS dan menanyakan lokasi *DataNode* yang bias diakses untuk menyimpan data tersebut. Setelah mendapatkan list nama dan alamat *DataNode* yang tersedia, komputer *client* akan langsung mentransfer data ke *DataNode* – *DataNode* tersebut. Data yang ditransfer tersebut secara otomatis dibelah menjadi block data berukuran 64 MB (sesuai pengaturannya). *Block* atau kepingan data inilah yang disimpan oleh tiap *DataNode*. Setelah *block* data tersimpan, tiap *DataNode* akan mengirimkan laporan kepada *NameNode* bahwa data sudah diterima dan disimpan. Ilustrasi penyimpanan dalam HDFS seperti ditunjukkan oleh Gambar 2.6 di bawah ini:



Gambar 2.6 : **Prosedur penyimpanan data dalam HDFS**

Sumber: (Wijaya, 2013)

Untuk pembacaan data, prosedur yang dilakukan hampir sama dengan prosedur menyimpan data. Pada saat *client* mengeksekusi perintah membaca data, computer *client* akan berkomunikasi dengan *NameNode* menanyakan nama dan alamat *DataNode* yang harus diakses untuk mendapatkan data yang diminta. Setelah informasi didapat, komputer *client* akan langsung mengakses *DataNode* yang bersangkutan untuk mendapatkan data yang diinginkan. Kemudian data akan ditampilkan pada tampilan komputer *client*. Ilustrasi pembacaan data dalam HDFS seperti ditunjukkan oleh Gambar 2.7 berikut:



Gambar 2.7 : **Prosedur pembacaan data dalam HDFS**

Sumber: (Wijaya, 2013)

2.2.5 Benchmarking

Dalam dunia komputasi, *benchmarking* dikenal sebagai tindakan menjalankan program komputer, seperangkat program, atau operasi lainnya, untuk menilai kinerja relatif suatu objek, biasanya dengan menjalankan sejumlah tes standar dan uji coba terhadapnya. Istilah '*benchmarking*' juga banyak digunakan untuk tujuan program evaluasi yang dirancang dengan cermat.

Benchmarking biasanya dikaitkan dengan penilaian karakteristik kinerja perangkat keras komputer, misalnya kinerja operasi floating point CPU, namun ada keadaan ketika teknik ini juga berlaku untuk perangkat lunak misalnya, *benchmarking* perangkat lunak untuk mengukur kinerja sistem manajemen basis data. *Benchmarking* juga dijadikan sebagai metode untuk membandingkan kinerja berbagai subsistem di berbagai chip / arsitektur sistem (Wiki, 2015). Evaluasi dengan *benchmarking* biasanya akan menghasilkan hasil akhir berupa angka (skor). Skor tersebut digunakan sebagai pembanding sistem mana yang memiliki performa lebih baik, tergantung dari parameter apa yang digunakan sebagai acuan untuk membandingkan kinerja sistem tersebut.

Melakukan aktivitas *benchmarking* memang tidaklah mudah, karena sering kali harus melibatkan melibatkan beberapa iteratif putaran untuk sampai pada kesimpulan yang dapat diprediksi dan berguna. Interpretasi data *benchmarking* juga tergolong sulit. Ada beberapa kriteria yang harus dimiliki agar hasil dari *benchmarking* dikatakan bagus (Ehliar, & Liu, 2004):

1. **Linearitas**

   Jika kinerja suatu mesin tiga kali lebih baik dari mesin lainnya, nilai tolak ukur untuk mesin pertama harus tiga kali lebih tinggi dari pada nilai mesin kedua.

2. Reliabilitas

Jika nilai suatu parameter untuk satu mesin lebih tinggi dari nilai parameter mesin lainnya, kinerja mesin pertama harus selalu lebih tinggi untuk dijadikan sebagai metrik acuan.

3. **Repeatbilitas**

Skor (nilai) dari parameter acuan harus sama apabila *benchmarking* diulang.

4. Independen

Aktivitas *benchmarking* tidak dipengaruhi oleh masukan dari produsen sistem yang ingin mengunggulkan bagian terbaik dari produk mereka.

Melakukan aktivitas *benchmarking* pada prosesor jaringan sedikit sulit dilakukan, karena pemrosesan jaringan bergantung secara inheren pada isi paket yang berasal dari jaringan itu sendiri. Hal ini membuat sulitnya menentukan parameter pengukuran yang adil dan sesuai. *Benchmarking* yang ada berkaitan dengan masalah ini dapat dilakukan dengan cara yang berbeda, misalnya dengan menjalankan *benchmark* untuk ukuran paket tertentu atau memvariasikan ukuran paket tertentu. Namun akan lebih baik apabila kinerja prosesor jaringan bisa diukur secara independen dari lalu lintas jaringan. Ada 2 pendekatan praktis yang dapat dipakai dalam melakukan aktivitas *benchmarking*, yakni *benchmarking* keseluruhan aplikasi dan *benchmarking* yang berfokus pada satu aspek dalam suatu aplikasi.

2.2.6 Testing Tools

Testing tools merupakan *tool*/alat/*software* yang akan digunakan sebagai pendukung dalam melakukan pengujian performasi dari kedua *file system*. Tool yang digunakan di masing-masing *file system* berbeda, tapi memiliki fungsi dan peran yang sama yakni digunakan sebagai pengukur kinerja I/O *file system*. Pada GlusterFS digunakan *tool* IOzone, sedangkan pada HDFS digunakan tool DFSIO.

2.2.6.1 IOzone

IOzone adalah alat pematokbandingan *(benchmarking)* yang berguna untuk menganalisis kinerja sistem file pada sejumlah platform yang berbeda, termasuk Linux, HP-UX, Solaris, dan banyak lainnya. Menggunakan sistem file I/O sebagai generasi beban utama, menghadirkan sistem yang diuji dengan berbagai macam permintaan berkas I/O, menjalankan berbagai file distribusi mulai dari yang berukuran kecil hingga yang sangat besar, ukuran *record* yang bervariasi saat melakukan sejumlah pola akses file yang berbeda. Hal inilah yang juga memungkinkan administrator *benchmark* untuk mengubah parameter uji sistem, mempengaruhi respon sistem untuk mendasari ukuran rute akses yang berbeda dan pola untuk menggambarkan masalah sistem yang mungkin tidak jelas, atau kemungkinan lain yang berbeda dari *platform* ke *platform*.

IOzone digunakan untuk menentukan analisis *file system* luas pada *platform* komputer *vendor*. Pengujian benchmark performa file I/O adalah dengan mengikuti beberapa *input* operasi berikut (www.iozone.org):

a) ***Write***: Tes mengukur kinerja dalam menulis file baru

b) ***Re-write***: Tes mengukur kinerja menulis file yang sudah ada.

c) ***Read***: Tes mengukur kinerja membaca file yang sudah ada.

d) ***Re-read***: Tes mengukur kinerja membaca file yang baru-baru baca.

e) ***Random read/write*** : Tes mengukur kinerja membaca/menulis file dengan akses yang dibuat secara acak dalam file

f) ***Backwards read*** : Tes mengukur kinerja membaca file *backward*

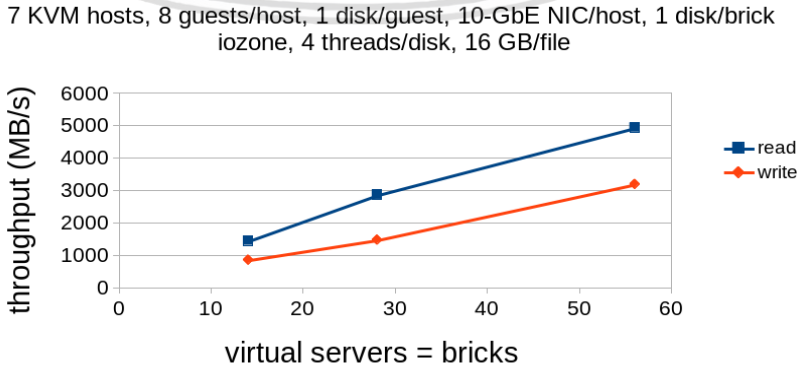
g) ***Record write*** : Tes mengukur kinerja menulis dan menulis ulang spot tertentu di alam file

h) ***Strided read*** : Tes mengukur kinerja membaca file dengan perilaku akses berkala

i) ***Fwrite*** : Tes mengukur kinerja dalam menulis file dengan fungsi library *fwrite()*

j) ***Fread*** : Tes mengukur kinerja dalam membaca file dengan fungsi library *fread()*

Dalam blog yang ditulis oleh Lambraight (2014) yang bertema "Pengujian skala output Gluster", telah dilakukan pengujian mengukur urutan skala *throughput read/write* file distribusi pada GlusterFS. *Tool* yang digunakan sebagai pematokbandingnya adalah IOzone, dan didapatkan hasil seperti Gambar 2.8 berikut:



Gambar 2.8 : **Hasil pengukuran skala urutan *throughput* proses *read/write* glusterfs dengan IOzone**

Sumber: (Lambright, 2014)

2.2.6.2 TestDFSIO

TestDFSIO adalah aplikasi *benchmarking* hadoop yang berfungsi untuk menguji kinerja I/O *(write/read)* dari HDFS. Aktivitas tersebut dilakukan dengan memanfaatkan kinerja dari *MapReduce* sebagai cara yang nyaman untuk membaca atau menulis file secara paralel. Setiap file yang dibaca atau ditulis dalam tugas (perintah) secara terpisah. Tes berjalan dengan banyak node, masing-masing menjalankan sebuah *thread* untuk menangani pembacaan dan penulisan. Tes ini disusun sebagai program *MapReduce* untuk menangani setiap *thread* secara simultan sebagai *map task* (pemetaan). Setiap *map task* menjalankan sebuah operasi dari tes, membuat atau membaca file. Setiap *map task* mencatat waktu dimulai dan selesainya operasi, ukuran data yang ditransfer, dan membagi ukuran tersebut setiap saat untuk menghitung kapasitas nilainya.

TestDFSIO terdapat dalam direktori */usr/hadoop-mapreduce/hadoop-mapreduce-client-jobclient-tests.jar*, serta dapat secara langsung dijalankan dengan mengeksekusi perintah:

```
$ hadoop jar /usr/lib/hadoop/hadoop-mapreduce/hadoop-
mapreduce-client-jobclient-*-tests.jar TestDFSIO -[argument]
```

Argument digunakan sebagai pilihan dari operasi TestDFSIO yang akan dijalankan dan harus disertakan di setiap eksekusi perintah operasi tes. Berikut beberapa pilihan *argument* yang digunakan dalam operasi tes (Lynch, 2017):

a) ***-write***: digunakan sebagai argumen untuk perintah pembuatan file

b) ***-read***: digunakan sebagai argumen untuk membaca file setelah menjalankan perintah *write.*

c) ***-clean***: menghapus semua data yang telah dibuat dengan TestDFSIO

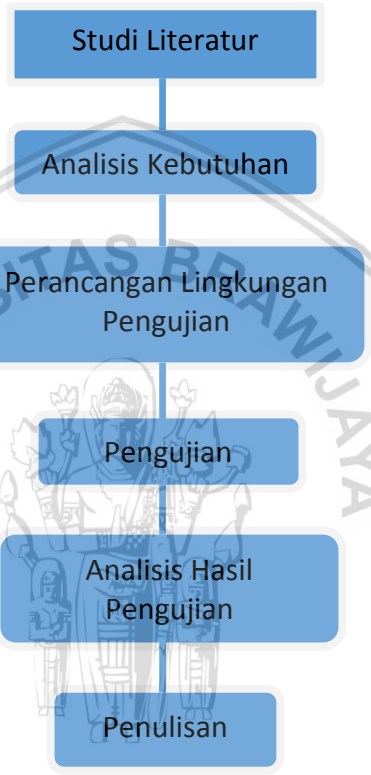
d) ***-nrFiles***: mendefinisikan banyaknya file yang digunakan dalam sebuah tes.

e) ***-fileSize***: mendefinisikan ukuran setiap file dari *map task* (nrFiles) dalam sebuah tes.

f) ***-resFile***: menunjukkan lokasi direktori dari hasil setiap tes yang dijalankan.

BAB 3 METODOLOGI PENELITIAN

Dalam bab ini akan dijelaskan langkah-langkah yang dilakukan untuk penyusunan penelitian "Analisis Perbandingan Kinerja File System GlusterFS dan HDFS Skenario Distribusi Striped dan Replicated" yang meliputi: studi literatur, analisis kebutuhan, perancangan lingkungan pengujian, pengujian, analisis hasil pengujian dan penulisan laporan. Gambar 3.1 berikut adalah diagram alir dari metode penelitian yang dilakukan.



Gambar 3.1 : Diagram Alur Metode Penelitian

Tipe penelitian yang dilakukan adalah non-implementatif analitik. Disebut demikian karena tujuan dilakukannya penelitian ini adalah menganalisa hasil dari perbandingan performa *file system*, GlusterFS dan HDFS, dalam melakukan aktivitas penyimpanan file terdistribusi yang menggunakan 3 host komputer (1 *host client* dan 2 *host server*) sebagai media pengujiannya.

3.1 Studi Literatur

Studi literatur mempelajari mengenai dasar teori yang digunakan untuk menunjang penulisan serta pelaksanaan penelitian ini. Teori-teori pendukung penulisan serta pemahaman tentang penelitian terkait diperoleh dari buku, jurnal, e-book, halaman website dan penelitian sebelumnya yang berkaitan tentang topik penelitian ini. Referensi utama yang diperlukan untuk menunjang penulisan ini adalah:

1. *Distributed File System* (Sistem File Terdistribusi)
2. *Cluster File*
3. GlusterFS
4. HDFS
5. *Test Tool*

3.2 Analisis Kebutuhan

Di dalam tahap ini akan dijelaskan mengenai segala kebutuhan perlengkapan yang diperlukan dalam penelitian ini. Kebutuhan yang diperlukan terdiri dari kebutuhan fungsional dan non-fungsional. Kebutuhan fungsional meliputi:

- Menjalankan fungsi *file system* GlusterFS dan HDFS
- Terbentuknya topologi GlusterFS dan HDFS
- Mampu melakukan kinerja penyimpanan file (*write/read*)

Kebutuhan non-fungsional yang diperlukan dalam penelitian ini terdiri dari kebutuhan perangkat keras dan kebutuhan perangkat lunak, yang dijabarkan sebagai berikut:

a. Kebutuhan perangkat keras, meliputi:
   - 2 unit PC dengan masing-masing spesifikasi minimal:
     - Processor Intel core i5 3 GHz, RAM 4 GB, Hardisk 1 TB.

b. Kebutuhan perangkat lunak, meliputi:
   - VirtualBox v5.1
   - Sistem operasi Ubuntu 14.04 LTS
   - *Gluster File System* (GlusterFS)
   - *Hadoop File System* (HDFS)
   - *Tools* (IOzone, DFSIO)
   - *Software Monitoring System* (Htop)

3.3 Perancangan Lingkungan Pengujian

Di dalam tahap ini akan dijelaskan mengenai perancangan ruang lingkup pengujian yang akan dilakukan untuk membandingkan kinerja dari GlusterFS dan HDFS dalam melakukan distribusi file. Perancangan lingkungan pengujian ini melibatkan sistem kerja dari Virtualbox, dengan pengujian sistem yang digunakan nantinya adalah model simulasi dari proses penyimpanan file terdistribusi pada topologi jaringan komputer yang divirtualisasikan.

Untuk perancangan lingkungan pengujian dalam jaringan komputer lokal secara virtual, maka digunakanlah aplikasi Virtualbox dengan memanfaatkan model jaringan *NAT Network*. *NAT Network* merupakan fitur pemodelan jaringan yang terdapat pada Virtualbox, di mana semua *virtual machine* dalam *host* yang sama dapat saling berkomunikasi antar satu sama lainnya (Oracle Coorporation, 2015). Di dalam Virtualbox ini akan dilakukan virtualisasi *host*, di mana host tersebut akan dibagi peran sebagai server (*slave*) dan klien (*master*) untuk melakukan aktivitas *write/read* file. Setiap host tersebut akan dipasang Ubuntu 14.04 LTS sebagai sistem operasinya.

Setelah dilakukannya virtualisasi host, maka setiap host akan diinstal aplikasi yang menjadi komponen utama dalam penelitian ini, yakni GlusterFS dan HDFS serta aplikasi pendukung untuk melakukan pengujian dan membandingkan keduanya dalam melakukan distribusi file, yakni IOzone dan DFSIO.

3.4 Pengujian

Setelah tahap perancangan ruang lingkup pengujian dan implementasi berhasil dilakukan, selanjutnya adalah tahap pengujian. Pengujian ini dilakukan dengan membandingkan kinerja GlusterFS dan HDFS ketika melakukan proses penyimpanan file terdistribusi di dalam lingkup jaringan komputer lokal. Pengujian akan dilakukan di dalam laboratorium multimedia, gedung E lantai 2, Fakultas Ilmu Komputer, Universitas Brawijaya.

Pengujian dilakukan dengan membandingkan parameter yang sudah ditentukan melalui skenario pengujian. Parameter pengujian tersebut adalah *throughput,* waktu eksekusi*,* beban CPU dan penggunaan memori, semua parameter tersebut akan diukur berdasarkan proses *write/read* file.

3.4.1 Skenario Pengujian

Skenario pengujian dalam penelitian ini dimaksudkan untuk memperoleh hasil dari parameter pengujian. Parameter pengujian yang dimaksud adalah nilai *throughput,* waktu eksekusi*, memory usage* dan beban kinerja CPU dari tiap-tiap node yang diuji. Skenario pengujian ini akan diterapkan di masing-masing topologi GlusterFS dan HDFS dengan kondisi yang sama. Proses read/write file dalam GlusterFS akan digunakan tool *iozone3*, sedangkan untuk HDFS akan digunakan DFSIO. Baik IOzone maupun DFSIO, keduanya merupakan tool bawaan masing-masing *file system* yang memang digunakan sebagai alat tolak ukur (*benchmarking*) kinerja dari *file system* tersebut.

3.4.1.1 Skenario Pengujian 1

Pada skenario pengujian 1 akan dilakukan replikasi file dari *node master* ke *node slave* dengan variasi ukuran file yang berbeda, yakni 1 GB, 2 GB dan 5 GB secara bergantian. Dimulai dengan menghubungkan direktori mount point dari *node master* ke *node slave*. Kemudian menjalankan proses *write/read file* dari direktori *node master* tersebut. Metode replikasi yang digunakan dalam pengujian ini akan membuat file yang ditulis/dibaca di *node master*, akan tereplikasi di kedua *node slave* dengan ukuran file yang sama.

3.4.1.2 Skenario Pengujian 2

Pada skenario pengujian 2 akan dilakukan replikasi file dari *node master* ke *node slave* dengan melihat variasi node yang terhubung. Skenario pengujian ini ditujukan untuk apakah jumlah node yang terhubung akan memberi perbedaan hasil pada parameter uji atau tidak. Dimulai dengan melakukan proses *write/read* file dengan 1 node (*node master*), 2 node (*node master* terhubung dengan 1 *node slave*) dan *multi node* (*node master* terhubung 2 *node slave*).

3.4.1.3 Skenario Pengujian 3

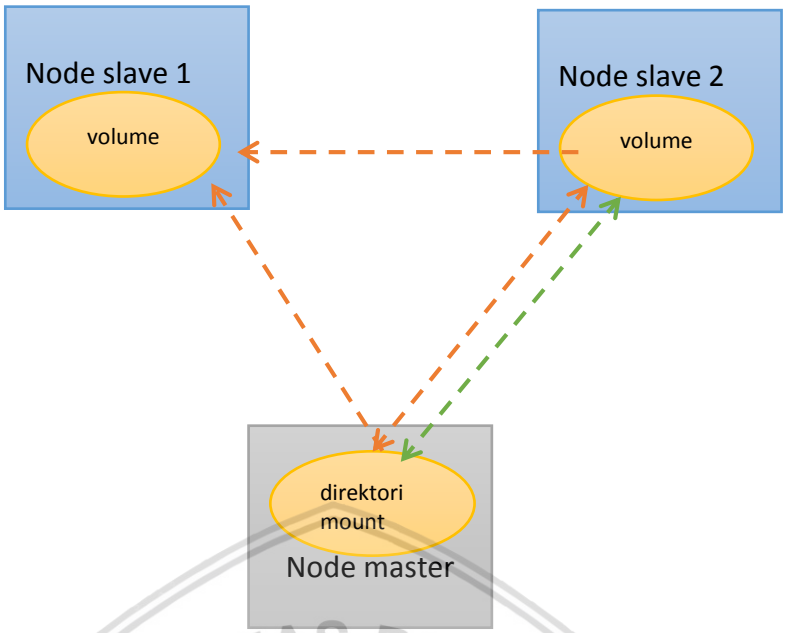
Skenario pengujian 3 akan dilakukan pengujian replikasi file dari *node master* ke *node slave* dengan melihat variasi ukuran block. Block adalah pecahan file yang berukuran kecil dengan satuan *byte*. Skenario pengujian ini ditujukan untuk mengetahui pengaruh ukuran block terhadap kinerja replikasi file yang dilakukan. Ukuran block yang diuji adalah 1 MB, 2 MB, 4 MB, 8 MB, 16 MB, dan 32 MB

3.4.1.4 Skenario Pengujian 4

Skenario pengujian 4 dilakukan dengan membuat beberapa *volume* dengan jumlah replikasi yang berbeda di tiap-tiap *volume*. Jumlah replikasi yang ditentukan adalah mulai dari 1 replikasi sampai 4 replikasi dengan kondisi *multi node*, yakni di mana *node master* terhubung dengan 2 *node slave* di masing-masing *file system*. Skenario pengujian ini dimaksudkan untuk mengetahui pengaruh jumlah replikasi terhadap kinerja proses replikasi file di tiap-tiap *volume* yang dibuat.

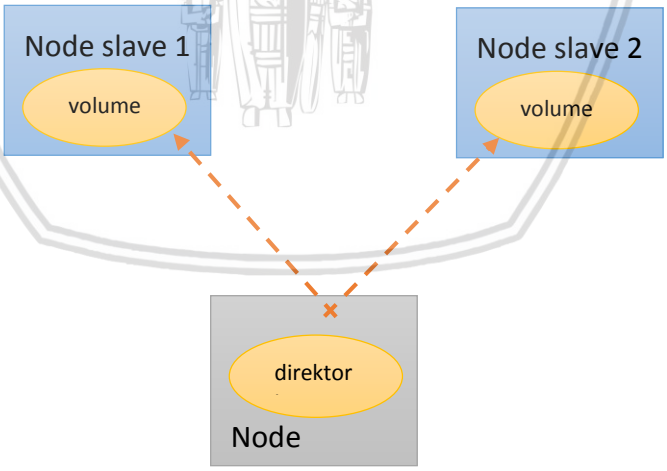
3.4.1.5 Skenario Pengujian 5

Skenario pengujian 5 merupakan skenario pengujian penanganan *fault* (kesalahan) yang terjadi pada *file system* ketikan melakukan proses *write/read* file masing-masing node. Ketika proses *write/read* file berlangsung akan dilakukan skenario penanganan *fault* (kesalahan) dengan mematikan 1 node *slave*. Ilustrasi skenario dapat dilihat pada Gambar 3.2 berikut:



Gambar 3.2 : Skenario *fault* dengan mematikan 1 node

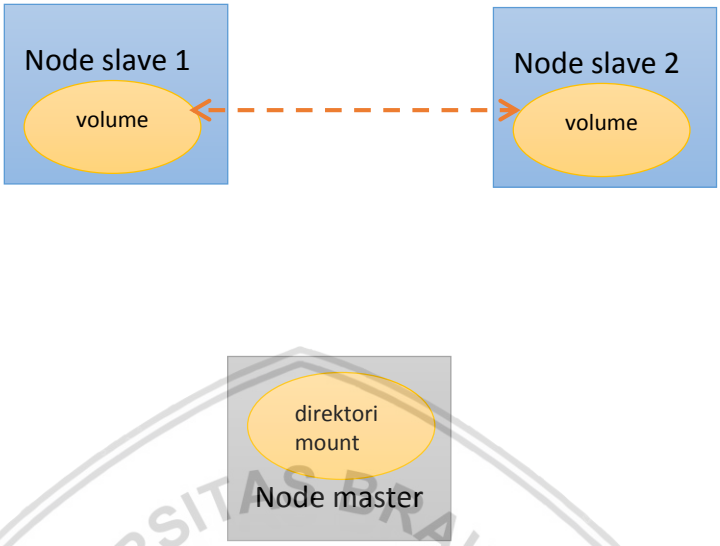
Kemudian setelah hasil didapatkan, proses *write/read* file diulangi kembali, kali ini ketika proses read/write berlangsung akan dilakukan pemutusan (mematikan) 2 node *slave* secara berurutan. Ilustrasi skenario dapat dilihat pada Gambar 3.3



Gambar 3.3 : Skenario *fault* dengan mematikan 2 node *slave*

Setelah hasil dari skenario tersebut didapat, proses read/write akan diulang kembali. Skenario penangan fault yang terakhir adalah ketika proses *write/read* file berlangsung, node master akan diputus (dimatikan). Skenario penanganan fault ini dilakukan untuk mengetahui apakah data yang direplikasi ke masing-

masing node salve dapat ditulis dan dibaca atau tidak. Ilustrasi skenario dapat dilihat pada Gambar 3.4



Gambar 3.4 : Skenario *fault* dengan mematikan node master

3.4.2 Throughput

Pada umumnya *throughput* merupakan kemampuan sebenarnya sebuah jaringan dalam melakukan pengiriman data. Nilai dari *throughput* didapatkan dengan menghitung jumlah data yang sukses dikirim dengan membagi waktu pengiriman data yang dibutuhkan. Dalam penelitian ini nilai *throughput* didapatkan dari besarnya data yang digunakan operasi tes membuat atau membaca file dibagi dengan waktu yang dibutuhkan dari setiap operasi tersebut.

3.4.3 Waktu Eksekusi

Waktu eksekusi merupakan sebuah mekanisme berbentuk task yang diberikan oleh CPU_*Time*, didefinisikan sebagai waktu yang dihabiskan oleh sistem dalam mengeksekusi task tersebut, termasuk waktu yang dibutuhkan untuk menjalankan *run-time* atau layanan sistem serupa. Mekanisme yang digunakan untuk mengukur waktu eksekusi adalah implementasi dari definisi task tersebut. Implementasi dijalankan berdasarkan task, apabila ada maka akan dibebankan oleh waktu eksekusi yang dikonsumsi oleh penanganan interupsi dan layanan *run-time.* Jenis CPU_*Time* mewakili waktu eksekusi sebuah *task*. Himpunan nilai dari jenis ini sesuai *one-to-one* dengan rentang bilangan bulat matematis yang didefinisikan oleh implementasi (Ada-Europe, 2011).

3.4.4 Beban CPU

Beban CPU adalah penggunaan kapasitas CPU oleh *device* dalam suatu waktu (CPU_*Time*). CPU_*Time* menghitung jumlah *clock ticks* atau *seconds.* Ketika beban CPU berada di atas 70%, pengguna akan merasakan *lag* (kinerja komputer melambat dan tersendat-sendat). Semakin tinggi beban CPU maka semakin tinggi pula *power* yang digunakan. Untuk menghindari hal tersebut maka CPU perlu diupgrade dari waktu ke waktu karena perkembangan teknologi membuat setiap *software* membutuhkan *resource* yang semakin besar untuk *user experience*, bila tidak maka pengguna harus melakukan "pengurangan" untuk menghindari terjadinya *lag*, seperti mengurangi resolusi atau *animations* pada game.

Beban CPU dapat dibagi menurut penggunaannya (Ehrhardt, 2010):

a) *User time* adalah waktu ketika CPU mengeksekusi *code* atau perintah dari *user space.*

b) *System time* adalah sejumlah waktu yang digunakan ketika CPU mengeksekusi *code* atau perintah di *kernel space.*

c) *Idle time* adalah sejumlah waktu ketika CPU sedang dalam mode *standby.*

d) *Steal time* terdapat pada *virtualized hardware,* adalah sejumlah waktu yang ingin dieksekusi oleh sistem operasi, namun tidak diizinkan oleh *hypervisor.*

3.4.5 Memory Usage

Memory usage adalah penggunaan RAM pada suatu *device.* Istilah ini digunakan untuk menunjukkan kapasitas penggunaan RAM selama aktivitas *user* dan *device* berlangsung. CPU membutuhkan RAM untuk menjalankan perintah secara singkat , ketika *device* menggunakan RAM melebihi kapasitas maka proses yang akan dieksekusi menjadi lambat (Ivan, 2016).

Untuk mendapatkan hasil berupa nilai dari parameter uji, maka digunakan pula aplikasi pendukung yakni, IOzone dan DFSIO.

3.5 Analisis Hasil

Pada tahap ini akan dilakukan analisis terhadap hasil pengujian yang sudah dilakukan dalam membandingkan kinerja GlusterFS dan HDFS. Sehingga dapat diperoleh hasil dan kesimpulan dari penelitian yang dilakukan.

Analisis hasil dilakukan dengan mengambil data dari hasil pengujian melalui skenario pengujian, yakni nilai *throughput,* waktu eksekusi, beban CPU dan *memory usage*. Data yang diambil dari keseluruhan parameter uji merupakan hasil dari pengamatan ketika melakukan proses *write/read file.* Hasil dari analisis kinerja keduanya akan menentukan manakah *file system* yang lebih efisien dalam melakukan proses penyimpanan file terdistribusi dalam lingkup jaringan komputer lokal.

3.6 Penarikan Kesimpulan

Penarikan kesimpulan dilakukan setelah semua tahapan perancangan, implementasi, pengujian dan analisis sistem terhadap metode yang digunakan mendapatkan hasil sesuai dengan tujuan penelitian. Penulisan saran berguna untuk memberikan pertimbangan atas hasil yang telah dilakukan dan untuk penelitian lanjutan ke depannya.

BAB 4 PERANCANGAN LINGKUNGAN PENGUJIAN

Pada bab ini akan dibahas tentang tahapan-tahapan dalam perancangan lingkungan pengujian yang digunakan untuk menguji kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *striped* dan *replicated*. Langkah-langkah perancangan tersebut akan mengacu pada perangkat keras dan perangkat lunak yang digunakan.

4.1 Instalasi dan Konfigurasi *File System*

Tahap pertama yang dilakukan dalam perancangan lingkungan pengujian ini adalah menyiapkan *host* atau komputer yang digunakan sebagai *node* dalam *file system*. Masing-masing *file system* terdiri dari 3 node, yakni 1 *node* sebagai master (*node master*) dan 2 node sebagai anak (*node slave*). Instalasi host dilakukan dengan memanfaatkan perangkat lunak VirtualBox v5.1 sebagai mesin virtual. Hal ini dilakukan dalam rangka untuk menghemat *resource* yang digunakan dan juga tujuan utama dalam penelitian ini adalah untuk menguji kinerja dari masing-masing *file system* itu sendiri.

Masing-masing virtual host yang telah dibuat akan diinstall sistem operasi Ubuntu 14.04 LTS. Spesifikasi masing-masing host dapat dilihat pada Tabel 4.1 berikut:

Tabel 4.1: Spesifikasi masing-masing *host file system*

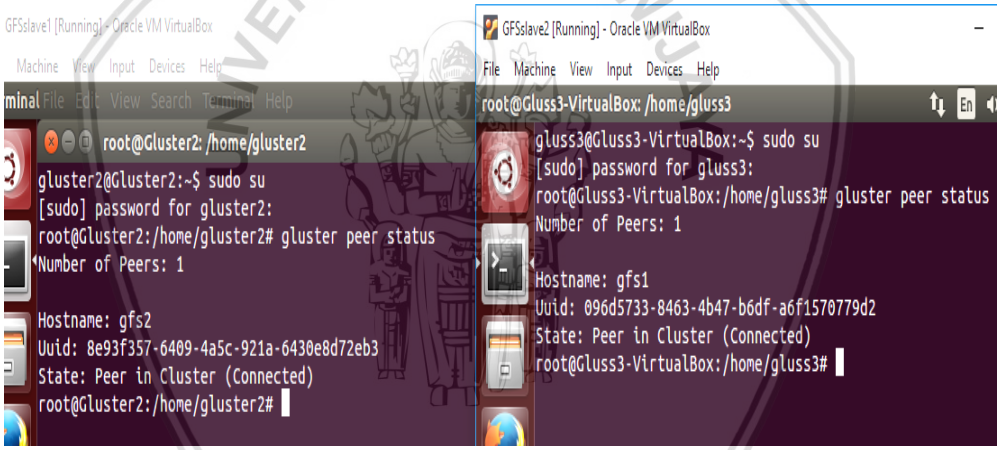
File system	GlusterFS	HDFS
Versi perangkat lunak	3.4.1	2.7.1
Sistem operasi	Ubuntu 14.04 LTS	Ubuntu 14.04 LTS
Jumlah total node	3	3
Node master	1	1
Kapasitas RAM	1 GB	1 GB
Kapasitas ruang penyimpanan	40 GB	40 GB
Node slave	2	2
Kapasitas RAM	@ 512 MB	@ 512 MB
Kapasitas ruang penyimpanan	@ 40 GB	40 GB

4.1.1 Instalasi dan Konfigurasi GlusterFS

Instalasi dilakukan pada ketiga node cluster GlusterFS. Pertama-tama lakukan instalasi pada kedua *node slave* terlebih dahulu. *Node slave* pada GlusterFS berperan sebagai *server*, di mana data nantinya akan disimpan dan

dibaca oleh *client.* Berikut adalah langkah-langkah dalam melakukan instalasi GlusterFS server:

1. Ubah dan definisikan *hostname* terlebih dahulu dengan perintah ***# sudo vi /etc/hosts***
2. *Update* terlebih dahulu *resource* sistem operasi dengan perintah ***# sudo apt-get update***
3. *Install* GlusterFS server di kedua node slave dengan perintah ***# sudo apt-get install glusterfs-server***
4. Jalankan perintah ***# glusterfs --version*** untuk mengecek bahwa GlusterFS sudah ter-*install* dan siap digunakan
5. Selanjutnya jalankan perintah ***gluster peer probe*** di salah satu *node server*. Perintah ini berfungsi untuk saling menghubungkan (sinkronisasi) kedua *node server*. Di sini perintah tersebut dijalankan di node server 1 ***# gluster peer probe gsf2.*** Kemudian jalankan perintah ***# gluster peer status*** untuk melihat bahwa kedua *node server* sudah terhubung dan menjadi *cluster*. Perhatikan Gambar 4.1 berikut:



Gambar 4.1 : Node server 1 dan server 2 terhubung menjadi cluster

6. Melakukan set-up GlusterFS *volume* dengan membuat sebuah direktori baru yang nantinya berfungsi sebagai *brick* pada *cluster server.* Selanjutnya jalankan perintah ***#gluster volume create [nama_volume] [tipe_volume] [jumlah_brick] transport [tcp \|rdma] [host:/direktori_brick] force***

```
# gluster volume create vol-Replikasi replica 2 transport tcp
gfs1:/replikasi1 gfs2:/replikasi2 force
```

Setelah muncul pesan sukses jalankan *volume* tersebut dengan perintah ***# gluster volume start [nama_volume]***

```
# gluster volume start vol-Replikasi
```

Cek status dan info dari *volume* yang dijalankan dengan perintah ***# gluster volume info*** dan ***# gluster volume status.***

7. Instalasi GlusterFS server selesai.

Berikutnya adalah langkah-langkah instalasi GlusterFS client. *Client* pada GlusterFS bertindak sebagai node master yang menjalankan perintah penulisan dan pembacaan file.

1. Jalankan perintah ***# sudo apt-get install glusterfs-client*** untuk menginstal GlusterFS *client*
2. Selanjutnya buat sebuah direktori baru yang akan digunakan *mount point* antara node master dan node slave.
3. Jalankan perintah ***# mount -t glusterfs [host:/volume] [mount_direktori]***

```
# mount –t glusterfs gfs1:/vol-Replikasi /tes-replikasi
```

Cek dengan perintah ***# df –h*** untuk melihat apakah *client* dan *server* sudah terhubung. Pada Gambar 4.2 di bawah ini, terlihat bahwa *mount point* pada *node client* GlusterFS telah terhubung dengan *volume* penyimpanan pada *node server*:

```
root@Gluster4-VirtualBox:/home/gluster4# df -h
Filesystem                 Size  Used Avail Use% Mounted on
udev                       491M  4,0K  491M   1% /dev
tmpfs                      101M  956K  100M   1% /run
/dev/mapper/ubuntu--vg-root  29G  5,7G   22G  21% /
none                       4,0K     0  4,0K   0% /sys/fs/cgroup
none                       5,0M     0  5,0M   0% /run/lock
none                       501M  152K  501M   1% /run/shm
none                       100M   40K  100M   1% /run/user
/dev/sda1                  236M   37M  187M  17% /boot
gfs1:/vol-Striped           61G   19G   39G  33% /mnt
gfs1:/vol-Replikasi         31G   11G   19G  36% /tes-replikasi
root@Gluster4-VirtualBox:/home/gluster4#
```

Gambar 4.2 : *Node client* dan *server* GlusterFS sudah terhubung

4. Installasi GlusterFS server selesai.

4.1.2 Instalasi dan Konfigurasi HDFS (Hadoop File System)

Instalasi dilakukan pada ketiga node cluster HDFS. Pertama-tama dilakukan instalasi pada node master terlebih dahulu. Langkah-langkah instalasinya adalah sebagai berikut:

1. Ubah dan definisikan hostname terlebih dahulu dengan perintah ***# sudo vi /etc/hosts***
2. Update terlebih dahulu *resource* sistem operasi dengan perintah ***# sudo apt-get update***
3. Memulai instalasi Hadoop dengan langkah-langkah sebagai berikut:

```
# sudo apt-get install default-jdk
# java –version
# sudo addgroup Hadoop
# sudo adduser –ingroup Hadoop hduser1
```

```
# sudo apt-get install ssh
# su hduser1
# ssh-keygen -t rsa -P ""
# cat $HOME/.ssh/id_rsa.pub >> $HOME/.ssh/authorized_keys
```

Download Hadoop dari *mirror* download apache lalu ekstrak konten-konten Hadoop ke lokasi yang dipilih.

```
# cd /usr/local
# sudo tar xzf hadoop-2.7.1.tar.gz
# sudo mv hadoop-2.7.1 hadoop
# sudo chown -R hduser1:hadoop hadoop
```

Proses instalasi paket Hadoop sudah selesai. Selanjutnya dilakukan konfigurasi pada beberapa file Hadoop dengan langkah-langkah sebagai berikut:

1. Memperbarui beberapa variabel *export* dengan menjalankan perintah ***# sudo gedit ~/.baschrc***. Tambahkan beberapa perintah *export* baru pada baris terakhir di dalam file ~/baschrc. Tujuan penambahan tersebut adalah untuk mempermudah proses pemanggilan dan ekseskusi perintah dengan Hadoop dan Java pada terminal. Beberapa *path* yang perlu ditambahkan adalah sebagai berikut:

```
# -- HADOOP ENVIRONMENT VARIABLES START -- #

export JAVA_HOME=/usr/lib/jvm/java-7-openjdk-i386
export HADOOP_HOME=/usr/local/hadoop
export PATH=$PATH:$HADOOP_HOME/bin
export PATH=$PATH:$HADOOP_HOME/sbin
export PATH=$PATH:/usr/local/hadoop/bin
export HADOOP_MAPRED_HOME=$HADOOP_HOME
export HADOOP_COMMON_HOME=$HADOOP_HOME
export HADOOP_HDFS_HOME=$HADOOP_HOME
export YARN_HOME=$HADOOP_HOME
export HADOOP_COMMON_LIB_NATIVE_DIR=$HADOOP_HOME/lib/native
export HADOOP_OPTS="-Djava.library.path=$HADOOP_HOME/lib"

# -- HADOOP ENVIRONMENT VARIABLES END -- #
```

Jalankan perintah ***# source ~/.baschrc*** untuk menerapkan perubahan yang sudah ditambahkan sebelumnya.

2. Selanjutnya melakukan konfigurasi pada file hadoop-env.sh dengan menjalankan perintah ***# sudo gedit /usr/local/hadoop/etc/hadoophadoop-***

env.sh. Ubah variabel *export* `JAVA_HOME=" "` menjadi `JAVA_HOME=/usr/lib/jvm/java-8-oracle.`

3. Langkah selanjutnya adalah menambahkan konfigurasi pada file core-site.xml, dengan menjalankan perintah ***# sudo /usr/local/hadoop/etc/hadoop/core-site.xml***. Konfigurasi yang ditambahkan pada file core-site.xml adalah sebagai berikut:

```
<property>
      <name>fs.default.name</name>
      <value>hdfs://hadoopmaster:9000</value>
</property>
```

4. Langkah selanjutnya adalah menambahkan konfigurasi pada file hdfs-site.xml, dengan menjalankan perintah ***# sudo /usr/local/hadoop/etc/hadoop/hdfs-site.xml***. Konfigurasi yang ditambahkan pada file hdfs-site.xml adalah sebagai berikut:

```
<property>
      <name>dfs.replication</name>
      <value>3</value>
</property>
<property>
      <name>dfs.namenode.name.dir</name>
      <value>file:/usr/local/hadoop_tmp/hdfs/namenode</value
      >

</property>
<property>
      <name>dfs.datanode.data.dir</name>
      <value>file:/usr/local/hadoop_tmp/hdfs/datanode</value
      >

</property>
```

5. Langkah selanjutnya adalah menambahkan konfigurasi pada file yarn-site.xml, dengan menjalankan perintah ***# sudo /usr/local/hadoop/etc/hadoop/yarn-site.xml***. Konfigurasi yang ditambahkan pada file yarn-site.xml adalah sebagai berikut:

```
<property>
   <name>yarn.nodemanager.aux-services</name>
   <value>mapreduce_shuffle</value>
```

```
</property>
<property>
   <name>yarn.nodemanager.aux-
   services.mapreduce.shuffle.class</name>
   <value>org.apache.hadoop.mapred.ShuffleHandler</value>
</property>
```

6. Terakhir adalah menambahkan konfigurasi pada file mapred-site.xml, dengan menjalankan perintah ***# sudo /usr/local/hadoop/etc/hadoop/mapred-site.xml***. Konfigurasi yang ditambahkan pada file mapred-site.xml adalah sebagai berikut:

```
<property>
      <name>mapreduce.framework.name</name>
      <value>yarn</value>
</property>
```

7. Selanjutnya jalankan perintah ***# sudo /usr/local/hadoop/etc/hadoop/slaves*** untuk mengisikan daftar *node* yang akan digunakan dalam lingkungan *file system* HDFS.

```
hadoopmaster
hnode1
hnode2
```

8. Setelah semua langkah diatas selesai dilakukan, matikan *node master*. Selanjutnya lakukan *clone* pada perangkat mesin virtual *hadoopmaster* tadi.
9. Selanjutnya jalankan perintah ***# sudo rm -r /usr/local/hadoop_tmp*** untuk menghapus direktori tersebut. Kemudian buat kembali direktori baru dengan perintah ***# sudo mkdir -p /usr/local/hadoop_tmp*.** Langkah ini bertujuan untuk menghapus *cache* dalam direktori tersebut.
10. Jalankan perintah ***# sudo chown -R hduser1 /usr/local/hadoop_tmp.*** Tujuannya agar direktori tersebut dapat diakses oleh semua *user.*
11. *Node master* HDFS siap dijalankan.

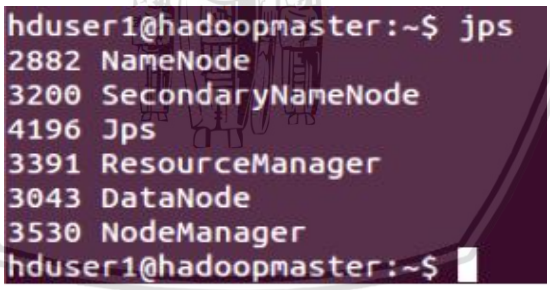
Berikutnya beralih ke instalasi HDFS *slave*. Pada HDFS yang berperan sebagai *slave* adalah *DataNode*, sedangkan peran *master* dijalankan oleh *NameNode*. Pada penelitian ini HDFS *slave* dibangun dari hasil *clone* mesin virtual *master*. Langkah-langkah dalam pembangunan *slave* adalah sebagai berikut:

1. Update file hdfs-site.xml dengan menjalankan perintah ***# sudo /usr/local/hadoop/etc/hadoop/hdfs-site.xml.*** Kemudian edit isi file seperti berikut ini:

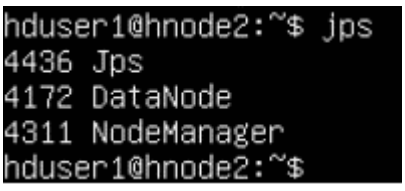
```
<property>
      <name>dfs.replication</name>
      <value>3</value>
</property>
<property>
      <name>dfs.datanode.data.dir</name>
      <value>file:/usr/local/hadoop_tmp/hdfs/datanode</value
      >

</property>
```

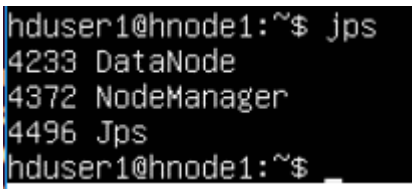
2. Selanjunya *reboot node slave* dengan menjalankan perintah ***# sudo reboot***.
3. Langkah serupa juga dilakukan pada *node slave* ke-2. Update konfigurasi pada file hdfs-site.xml, setelah selesai *reboot node slave* ke-2.
4. Kemudian lakukan langkah ke 9 dan 10 pada instalasi *node master* HDFS.
5. *Node slave* HDFS siap dijalankan.

Setelah instalasi *node master* dan *node slave* selesai, berikutnya adalah menjalankan modul *file system* HDFS. Jalankan perintah ***# hdfs namenode -format*** pada *node master* HDFS. Kemudian jalankan perintah ***# start-all.sh***, perintah tersebut akan menjalankan seluruh modul HDFS, cek dengan perintah ***# jps*** pada setiap *node*. Apabila menunjukkan hasil yang sama seperti Gambar 4.3 dan Gambar 4.4, itu menunjukkan bahwa modul *file system* HDFS sudah berjalan sepenuhnya.



Gambar 4.3 : Modul *node master* berjalan





Gambar 4.4 : Modul *node slave* 1 dan *node slave 2* berjalan

4.2 Menjalankan Fungsi *File System*

Setelah langkah instalasi dan konfigurasi masing-masing *file system* selesai, tahap kedua dalam perancangan lingkungan pengujian adalah implementasi *file system*. Pada tahap ini akan dilakukan uji coba kinerja *system* dengan menjalankan operasional penulisan (*write*) dan pembacaan (*read*) file.

4.2.1 Menjalankan Fungsi GlusterFS *Replicate*

a) Jalankan semua *node* pada modul GlusterFS

b) Buka terminal lalu masuk sebagai *root.*

c) Cek koneksi antara *node server* dengan perintah ***# gluster peer status*.** Pastikan kedua *node server* berstatus *connected.*

d) Buat sebuah direktori baru dimasing-masing *node server* (contoh : gfs1:/replikasi1, gfs2:/replikasi2)

e) Pada salah satu *node server* jalankan perintah

```
# gluster volume create vol-Replikasi replica 2 transport
tcp gfs1:/replikasi1 gfs2:/replikasi2 force
```

f) Jalankan *volume* yang baru saja dibuat dengan perintah

```
# gluster volume start vol-Replikasi
```

g) Informasi *volume* tersebut dapat diperiksa dengan menjalankan perintah ***# gluster volume info vol-Replikasi*.** Informasi dari *volume* tersebut seperti ditunjukkan pada Gambar 4.5

```
Volume Name: vol-Replikasi
Type: Replicate
Volume ID: b1951e0a-0b22-4a5c-a385-f73d91670346
Status: Started
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: gfs1:/replikasi1
Brick2: gfs2:/replikasi2
```

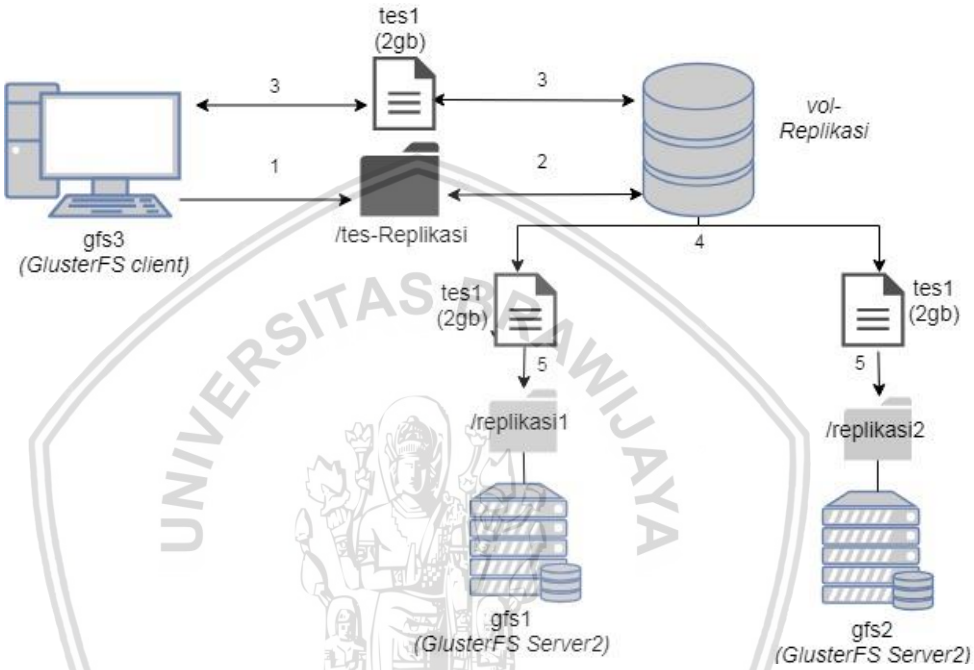
Gambar 4.5 : *Gluster volume info* (*replicate*)

h) Pada *node client* install iozone dengan perintah ***# apt-get install iozone3*.** IOzone digunakan sebagai *tool* untuk operasi penulisan dan pembacaan *file* di GlusterFS.

i) Install *software monitoring system*, di dalam penelitian ini perangkat yang digunakan adalah htop.

j) Buat direktori baru pada *node client* (***# mkdir /tes-replikasi***) sebagai *mount point* antara *node client* dan *node server.*

k) Ilustrasi operasional *write/read file* seperti yang diperlihatkan pada Gambar 4.6. Pada *node client,* (1) dijalankan perintah ***# mount –t glusterfs gfs1:/vol-Replikasi /tes-replikasi.*** Perintah tersebut akan menghubungkan direktori *mount point* pada GlusterFS *client* dengan direktori *brick* /replikasi1 dan /replikasi2 di kedua *node server* (gfs1, gfs2), kemudian menjadi satu-kesatuan ruang penyimpanan dalam *volume* vol-Replikasi (2).



Gambar 4.6: Ilustrasi operasional *write/read file* GlusterFS *replicated*

l) Melakukan operasional penulisan dan pembacaan *file* (3) dengan menjalankan perintah:

```
# iozone -I -R -b /tes-replikasi/report.xls -c -e -i 0 -c
-e -i 1 -M -+n -s 2g -r 16m -t 1 -+u -Q -w -F /tes-
replikasi/tes1
```

Keterangan :

- `-I -R -b /tes-replikasi/report.xls` : informasi tentang aktivitas penulisan/pembacaan file disimpan dalam direktori tes-replikasi, file report.xls
- `-c -e -i 0` : menunjukkan perintah *write file*
- `-c -e -i 1` : menunjukkan perintah *read file*
- `-s 1g` : file berukuran 1 Gigabyte
- `-t 1 -+u` : *thread file* berjumlah 1
- `-Q -w -F /tes-replikasi/tes1` : *file output* disimpan di direktori tes-replikasi dengan *label* tes1.

m) Modul GlusterFS *replicate* membuat file yang dieksekusi dengan perintah *write/read file*, direplikasi (duplikasi) menjadi 2 buah file dengan ukuran yang sama (4) serta tersimpan di masing-masing *node server* di dalam direktori *brick* /replikasi1 dan /replikasi2 (5).

n) Berikut adalah hasil dari operasional *write/read file* pada GlusterFS *replicate* ditunjukkan oleh Gambar 4.7, Gambar 4.8 dan Gambar 4.9

Gambar 4.7 : Hasil pada *node client* GlusterFS *replicate*

Gambar 4.8 : Hasil pada *node server gfs1* GlusterFS *replicate*

Gambar 4.9 : Hasil pada *node server gfs2* GlusterFS *replicate*

4.2.2 Menjalankan Fungsi GlusterFS *Stripe*

a) Membuat direktori baru pada masing-masing *node server* GlusterFS (gfs1:/stripe-s1, gfs2:/stripe-s2)

b) Pada salah satu *node server* jalankan perintah:

```
# gluster volume create vol-Striped stripe 2 transport tcp
gfs1:/stripe-s1 gfs2:/stripe-s2 force
```

c) Perintah tersebut berfungsi untuk membuat sebuah *volume* baru di GlusterFS bernama vol-Striped. *Volume* tersebut bertipe distribusi *stripe* dengan 2 *brick* (jumlah direktori penyimpanan *cluster*) yaitu direktori stripe-s1 di *node* gfs1 dan direktori stripe-s2 di *node* gfs2.

d) Jalankan *volume* yang baru saja dibuat dengan perintah:

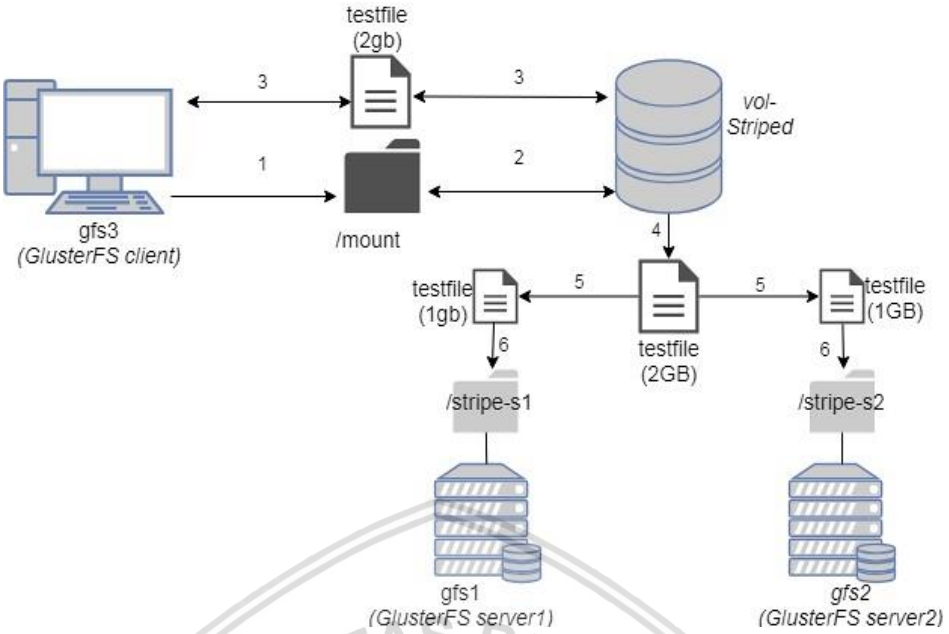
```
# gluster volume start vol-Striped
```

e) Informasi *volume* tersebut dapat diperiksa dengan menjalankan perintah ***# gluster volume info vol-Striped***. Informasi dari *volume* tersebut seperti ditunjukkan pada Gambar 4.10

```
Volume Name: vol-Striped
Type: Stripe
Volume ID: d608b318-a8dd-4a93-a7a9-6d97f6ada54e
Status: Started
Number of Bricks: 1 x 2 = 2
Transport-type: tcp
Bricks:
Brick1: gfs2:/stripe-s2
Brick2: gfs1:/stripe-s1
```

Gambar 4.10 : *Gluster volume info (striped)*

f) Ilustrasi operasional *write/read file* pada GlusterFS *striped* diperlahatkan oleh Gambar 4.11. Menghubungkan *node server* dengan *node client* melalui direktori /mnt, dengan menjalankan perintah ***# mount –t glusterfs gfs1:/vol-Striped /mnt*** pada *node client* (1). Perintah tersebut akan menghubungkan direktori *mount point* pada GlusterFS *client* dengan direktori *brick* /stripe-s1 *node server* gfs1 dan /stripe-s2 di *node server* gfs2, kemudian menjadi satu-kesatuan ruang penyimpanan dalam *volume* vol-Striped (2).



Gambar 4.11 : Ilustrasi operasional *write/read file* GlusterFS *striped*

g) Melakukan operasional penulisan dan pembacaan *file* (3) dengan menjalankan perintah:

```
# iozone -I -R -b /mnt/report.xls -c -e -i 0 -c -e -i 1 -
M -+n -s 2g -r 16m -t 1 -+u -Q -w -F /mnt/testfile
```

Modul GlusterFS *replicate* membuat sebuah file yang dieksekusi dengan perintah *write/read file* (4), dipecah (dibagi) menjadi 2 buah file dengan ukuran setengah dari ukuran file yang semula (5). Ukuran file yang dipecah terbagi sesuai dengan jumlah brick yang dideskripsikan saat pembuatan *striped volume*. Masing-masing pecahan file akan disimpan pada setiap *brick*. Pada Gambar 4.11, pecahan file tersimpan di masing-masing *node server* di dalam direktori *brick* /stripe-s1 dan /stripe-s2 (6).

h) Berikut adalah hasil dari operasional penulisan dan pembacaan file pada GlusterFS *stripe* ditunjukkan oleh Gambar 4.12, Gambar 4.13 dan Gambar 4.14

```
root@Gluster4-VirtualBox:/home/gluster4# du -h /mnt
1,1G    /mnt
```

Gambar 4.12 : Hasil pada *node client* GlusterFS *stripe*

Gambar 4.13 : Hasil pada *node server gfs1* GlusterFS *stripe*

Gambar 4.14 : Hasil pada *node server gfs2* GlusterFS *stripe*

4.2.3 Menjalankan Fungsi HDFS

Khusus pada *file system* HDFS, mode replikasi dan stripe berjalan bersamaan sebagai satu kesatuan. Replikasi merupakan aktifitas penyimpanan file yang didistribusikan ke dalam *datanode*. File-file yang disimpan dipecah menjadi *block file*, di mana setiap *block file* disimpan secara random *(stripe)* di dalam setiap *node.*

a) Jalankan semua *node* pada HDFS *(node slave, node master).*

b) Masuk sebagai hduser1

c) Jalankan perintah ***# start-all.sh*** pada *node master* HDFS. Cek keseluruhan fungsional modul HDFS dengan perintah ***# jps.***

d) Install *software monitoring system*, *software* yang digunakan sama seperti pada GlusterFS yaitu htop.

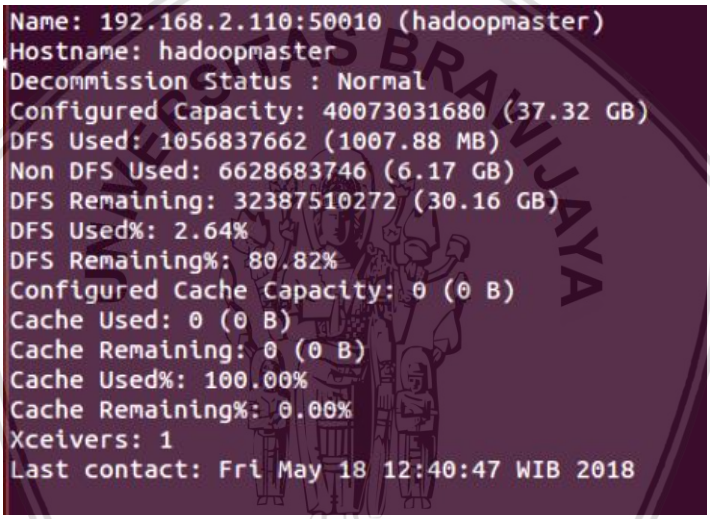
e) Mulai menjalankan operasional penulisan dan pembacaan file. *Tool* yang digunakan adalah DFSIO. DFSIO merupakan *benchmark tool* bawaan Hadoop yang memamng digunakan untuk mengukur kinerja *file system* Hadoop. Perintah yang dijalankan adalah

```
# hadoop jar /usr/local/hadoop/share/hadoop/mapreduce/
hadoop-mapreduce-client-jobclient-2.7.1-tests.jar
TestDFSIO –write –nrFiles 1 –fileSize 1000MB
```

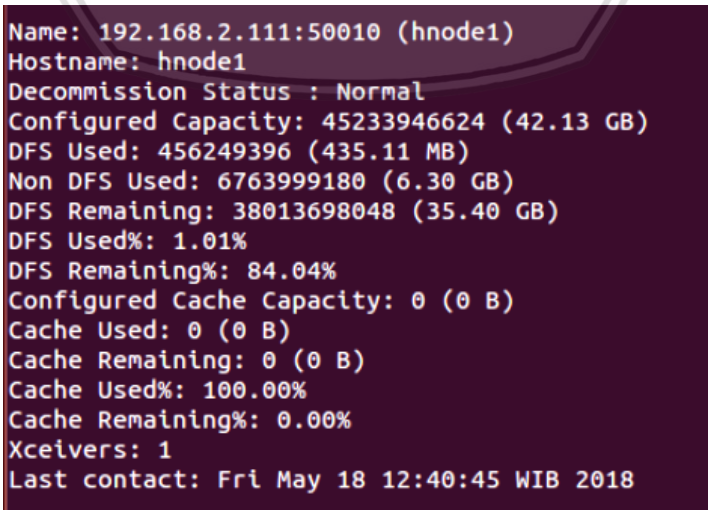
Keterangan

- `/usr/....-jobclient-2.7.1-tests.jar TestDFSIO` : menjalankan fungsi *tool* DFSIO
- `–write` : argumen ini memberitahukan pada *tool* DFSIO untuk melakukan operasional *write file*
- `–nrFiles 1` : argumen yang menunjukkan jumlah/banyaknya file
- `–fileSize 1000MB` : argument yang digunakan untuk menentukan besarnya ukuran file

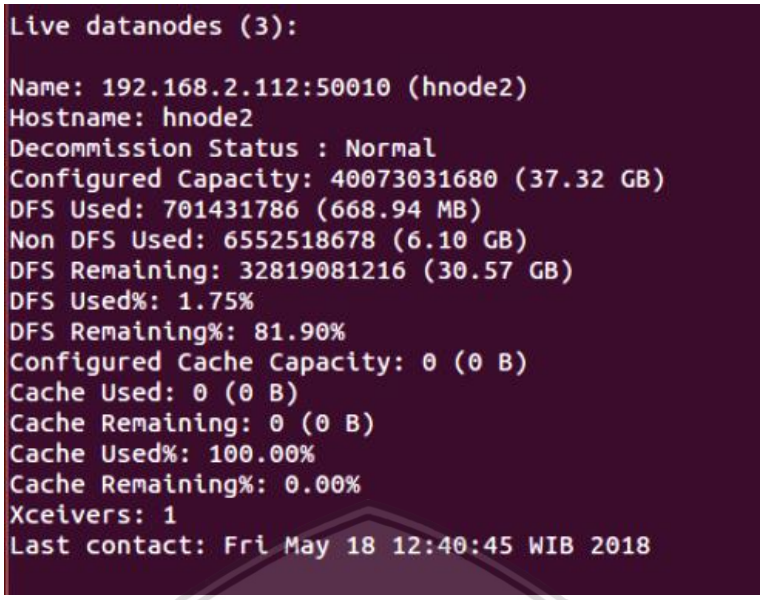
f) Berikut adalah hasil dari operasional *write/read* file pada HDFS ditunjukkan oleh Gambar 4.15, Gambar 4.16 dan Gambar 4.17



Gambar 4.15 : Hasil operasional *write/read file node master*



Gambar 4.16 : Hasil operasional *write/read file node slave1*



Gambar 4.17 : Hasil operasional *write/read file node slave2*

BAB 5 PENGUJIAN DAN ANALISIS DATA

Pada bab 5 akan dijelaskan mengenai pengujian dan analisis data dari hasil pengujian yang diperoleh. Pengujian dilakukan dengan menjalankan masing-masing skenario pengujian yang sudah ditentukan. Data yang diperoleh akan dianalisis berdasarkan parameter pengujiannya (*throughput,* waktu eksekusi, beban penggunaan CPU dan *memory usage*). Kriteria yang dijadikan acuan adalah apabila nilai *throughput* semakin besar maka operasional *write/read file* menjadi lebih cepat (waktu lebih cepat) dan sebaliknya apabila nilai *throughput* kecil maka operasional *write/read file* menjadi lama (waktu lebih lama).

5.1 Pengujian Skenario Pertama : Variasi Ukuran File

Pengujian skenario yang pertama kali dilakukan adalah pengujian variasi ukuran file. Pada masing-masing *file system* akan dilakukan operasi penulisan dan pembacaan file dengan ukuran 1 GB, 2 GB dan 5 GB, yang masing-masing dilakukan sebanyak 5 kali. Parameter yang diukur pada pengujian skenario ini adalah *throughput,* waktu eksekusi, beban penggunaan CPU dan *memory usage*. Prosedur pengujian variasi ukuran file adalah sebabagai berikut:

- Pastikan bahwa semua *node (node master, node slave)* sudah terhubung
- Jalankan perintah operasional penulisan file dan pembacaan file, dimulai dengan ukuran file sebesar 1 GB, 2 GB dan yang terakhir 5 GB.
- Setiap operasional penulisan dan pembacaan file diulang sebanyak 5 kali
- Hasil keseluruhan operasional penulisan dan pembacaan dimasukkan ke dalam tabel dan kemudian dilakukan analisis hasil untuk perbandingan kinerja *file system*

5.1.1 Hasil Pengujian dan Analisis Data

Hasil pengujian skenario pertama dengan operasional penulisan dan pembacaan file yang dilakukan sebanyak 5 kali di masing-masing *file system*. Hasil yang diperoleh merupakan nilai rata-rata dari setiap paramater pengujian ketika dilakukan operasional *write/read* file dengan ukuran file 1 GB, 2 GB dan 5 GB. Hasil operasional *write/read file* pada GlsuterFS-*Stripe/Replicate* diperlihatkan oleh Tabel 5.1 dan Tabel 5.2 di bawah ini:

Tabel 5.1: Hasil rata-rata nilai parameter pengujian skenario pertama GlusterFS-*Stripe*

Parameter Uji	Operasional *File System*	
	write	*read*
Throughput (Mbps)	59.33	43.04
Waktu Eksekusi (s)	48.85	68.66
Beban Penggunaan CPU (%)	49.80	65.13
Memory Usage (%)	3.4	6.53

Tabel 5.2 : Hasil rata-rata nilai parameter pengujian skenario pertama GlusterFS-*Replicate*

Parameter Uji	Operasional *File System*	
	write	*read*
Throughput (Mbps)	30.16	70.30
Waktu Eksekusi (s)	85.37	41.38
Beban Penggunaan CPU (%)	52.53	40.67
Memory Usage (%)	2.92	6.2

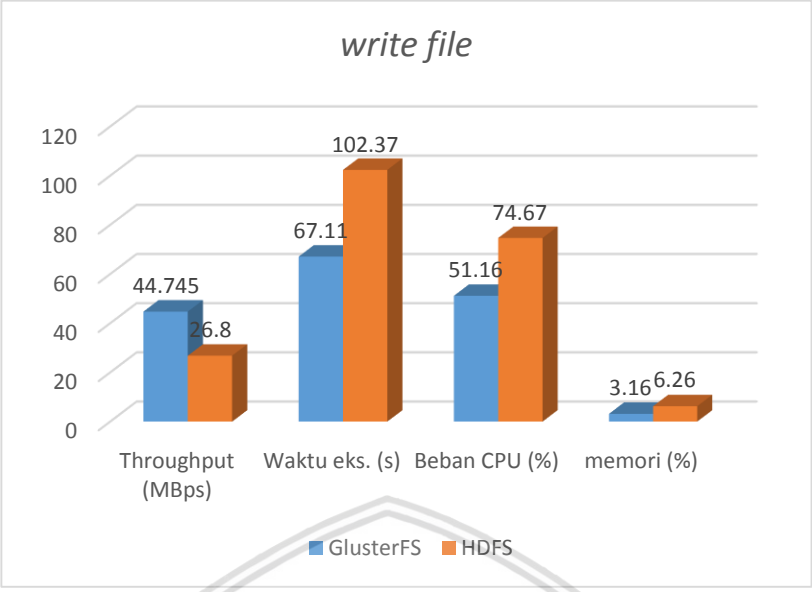
Sedangkan hasil operasional *write/read file* HDFS dapat dilihat dalam Tabel 5.3 di bawah ini:

Tabel 5.3 : Hasil rata-rata nilai parameter pengujian skenario pertama HDFS

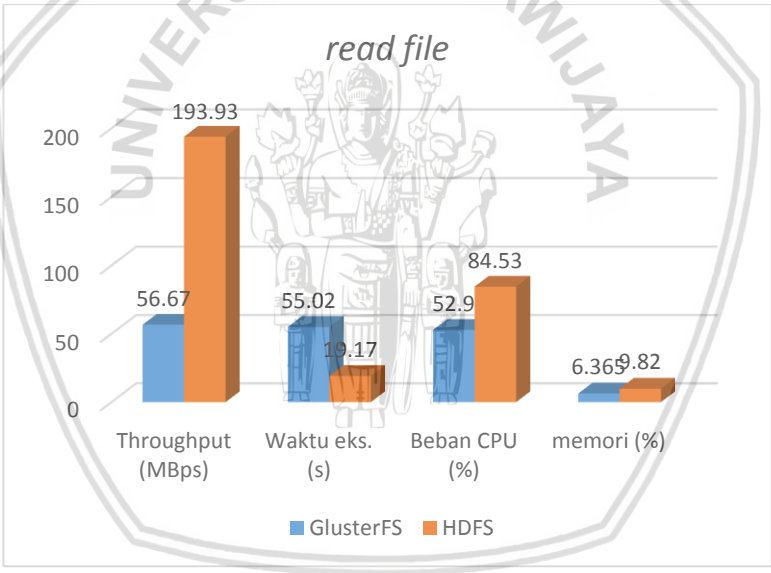
Parameter Uji	Operasional *File System*	
	write	*read*
Throughput (Mbps)	26.80	193.93
Waktu Eksekusi (s)	102.37	19.17
Beban Penggunaan CPU (%)	74.67	84.53
Memory Usage (%)	6.26	9.82

Setelah dilakukan analisis hasil pengujian skenario pertama terhadap kinerja *file system*, diketahui bahwa GlusterFS lebih cepat dalam melakukan operasional *write file* dengan rata-rata *throughput* sebesar 44,74 MBps dan waktu eksekusi selama 67,11 detik. Sedangkan pada HDFS didapatkan hasil rata-rata *throuhput* sebesar 26,80 MBps dan waktu eksekusi selama 102,37 detik. Namun saat operasional *read file* dilakukan HDFS menunjukkan kinerja yang lebih cepat dengan rata-rata *throuhput* sebesar 193,93 MBps dan waktu eksekusi selama 19,17 detik. Sedangkan pada GlusterFS didapatkan rata-rata *throuhput* sebesar 56,67 MBps dan waktu eksekusi selama 55,02 detik.

Analisis konsumsi daya saat melakukan operaional *write/read file* memperlihatkan hasil HDFS dengan beban penggunaan CPU 25% lebih tinggi daripada GlusterFS dan penggunaan memori yang 4% sedikit lebih tinggi. Hasil dari pengujian skenario pertama dapat disimpulkan bahwa GlusterFS memiliki kinerja yang lebih ringan daripada HDFS, seperti yang diperlihatkan pada Gambar 5.1 dan Gambar 5.2



Gambar 5.1 : Grafik perbandingan kinerja GlusterFS dan HDFS pengujian skenario pertama *write file*



Gambar 5.2 : Grafik perbandingan kinerja GlusterFS dan HDFS pengujian skenario pertama *read file*

5.2 Pengujian Skenario Kedua : Variasi *Node* (*Single Node – Multi Node*)

Pengujian skenario yang kedua adalah pengujian variasi jumlah node yang terhubung *(single node – multi node).* Dimulai dengan melakukan proses *write/read*file dengan *single node* (*node master* terhubung dengan 1 *node slave*) dan *multi node* (*node master* terhubung 2 *node slave*). Tujuan dari pengujian skenario kedua ini adalah untuk membandingkan kinerja kedua *file system*

berdasarkan parameter pengujian beban penggunaan CPU dan *memory usage.* Prosedur pengujian variasi node adalah sebagai berikut:

- Menghubungkan *node master* dengan salah satu *node slave* untuk pengujian *single node.*
- Menghubungkan *node master* dengan 2 *node slave* untuk pengujian *multi node.*
- Menjalankan perintah operaional penulisan dan pembacaan file berukuran 2 Gb sebanyak 3 kali pada masing-masing variasi *node.*
- Hasil dari pengujian tersebut dimasukkan ke dalam tabel untuk selanjutnya dilakukan analisis perbandingan kinerja *file system.*

5.2.1 Hasil Pengujian dan Analisis Data

Hasil pengujian skenario kedua yaitu variasi jumlah *node* yang dilakukan sebanyak 3 kali di masing-masing kondisi *single node* dan *multinode.* Hasil pengujian mode *single node* dapat dilihat pada Tabel 5.4, dalam operasional *write file* dan Tabel 5.5, dalam operasional *read file*.

Tabel 5.4 : Hasil pengujian skenario *single node write file*

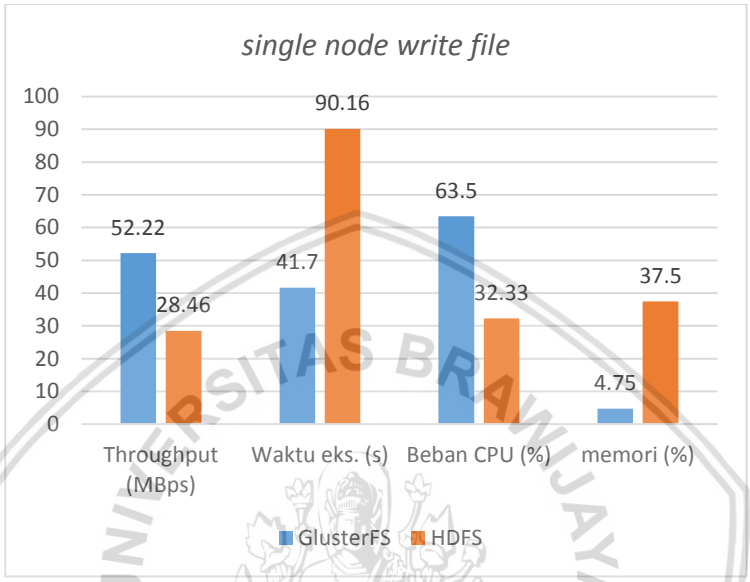
Parameter Uji	GlusterFS-*Stripe*	GlusterFS-*Replicate*	HDFS
Throughput (Mbps)	61.78	42.65	28.46
Waktu Eksekusi (s)	33.96	49.44	90.16
Beban Penggunaan CPU (%)	58	69	32.33
Memory Usage (%)	3	6.50	37.50

Tabel 5.5 : Hasil pengujian skenario *single node read file*

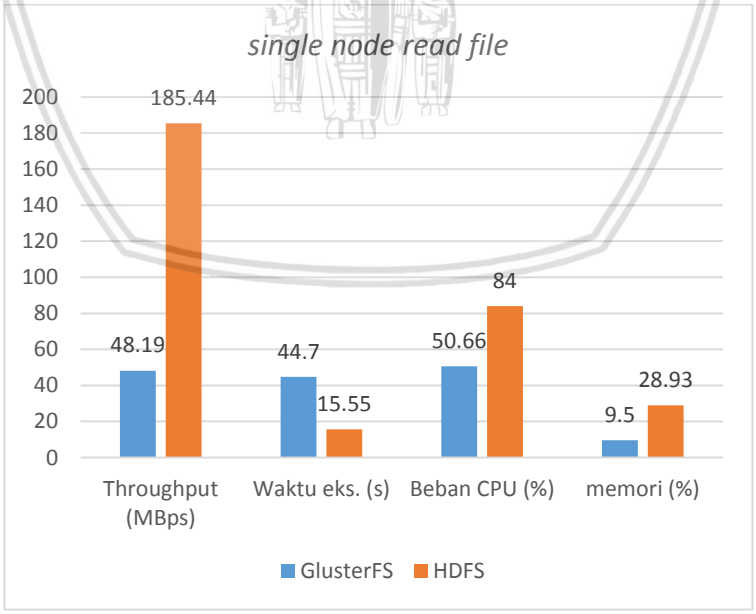
Parameter Uji	GlusterFS-*Replicate*	GlusterFS-*Stripe*	HDFS
Throughput (Mbps)	54.84	41.54	185.44
Waktu Eksekusi (s)	38.77	50.64	15.55
Beban Penggunaan CPU (%)	57.33	44	84
Memory Usage (%)	3.17	6.33	28.93

Hasil pengujian *single node* pada Tabel 5.4 menunjukkan bahwa GlusterFS melakukan kinerja lebih cepat dan lebih stabil dibandingkan HDFS. Pada operasional *write file* GlusterFS menunjukkan rata-rata *throughput* sebesar 52,22 MBps, waktu eksekusi selama 41,7 detik, beban penggunaan CPU sebesar 63,5% dan penggunaan memori sebesar 4,75%. Sedangkan pada HDFS menunjukkan rata-rata *throughput* sebesar 28,46 MBps, waktu eksekusi selama 90.16 detik beban penggunaan CPU sebesar 32.33% dan penggunaan memori sebesar 37,50%.

Pada operasional *read file* GlusterFS menunjukkan rata-rata *throughput* sebesar 48,19 MBps, waktu eksekusi selama 44,7 detik beban penggunaan CPU sebesar 50,66% dan penggunaan memori sebesar 9,5%. Sedangkan pada HDFS menunjukkan rata-rata *throughput* sebesar 185,44 MBps, waktu eksekusi selama 15,55 detik beban penggunaan CPU sebesar 84% dan penggunaan memori sebesar 28,93%. Hasil perbandingan kinerja *file system* mode *single node* dapat dilihat pada Gambar 5.3 dan Gambar 5.4 berikut:



Gambar 5.3 : Grafik perbandingan kinerja mode *single node write file*



Gambar 5.4 : Grafik perbandingan kinerja mode *single node read file*

Untuk hasil pengujian *file system* mode *multi node* dapat dilihat pada Tabel 5.6 dan Tabel 5.7 berikut:

Tabel 5.6 : Hasil pengujian skenario *multi node write file*

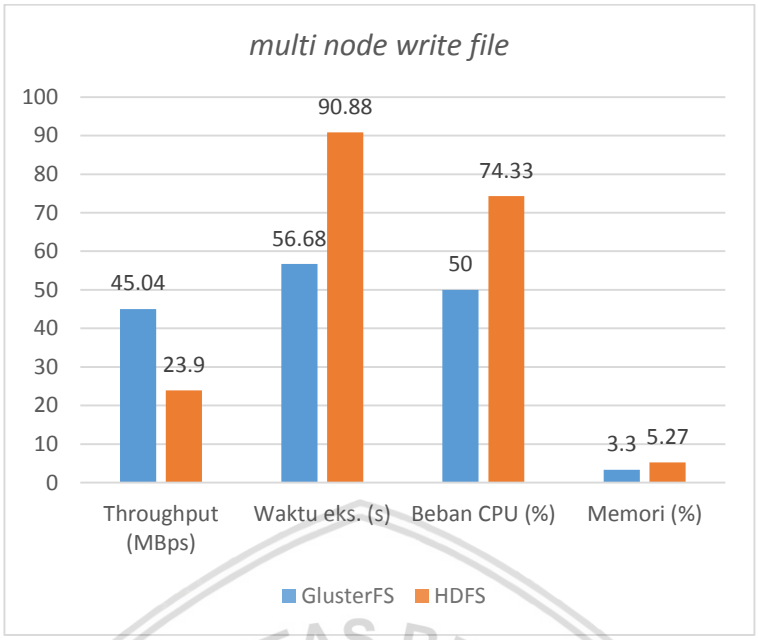
Parameter Uji	GlusterFS-*Replicate*	GlusterFS-*Stripe*	HDFS
Throughput (Mbps)	27.15	62.92	23.90
Waktu Eksekusi (s)	77.96	35.39	90.88
Beban Penggunaan CPU (%)	51.33	48.67	74.33
Memory Usage (%)	3	3.6	5.27

Tabel 5.7 : Hasil pengujian skenario *multi node read file*

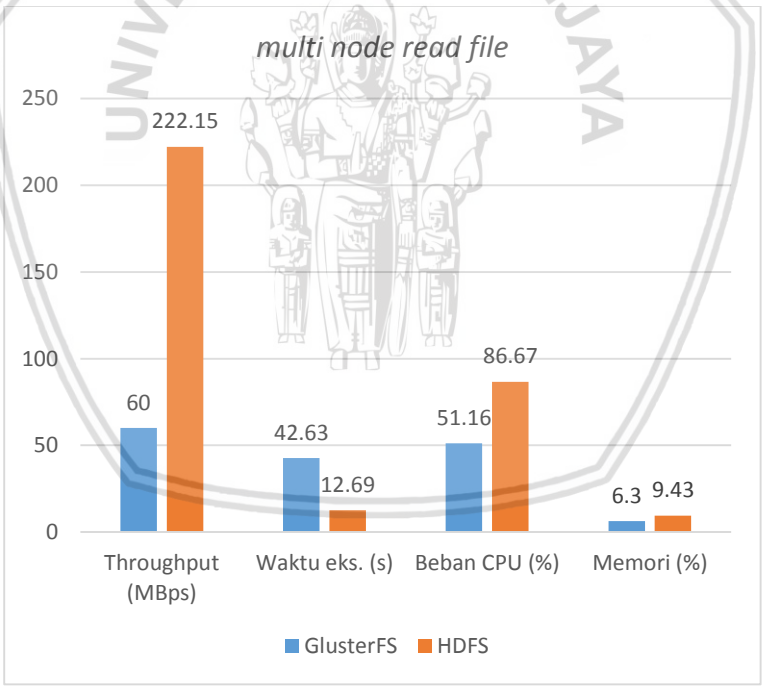
Parameter Uji	GlusterFS-*Replicate*	GlusterFS-*Stripe*	HDFS
Throughput (Mbps)	74.91	45.10	222.15
Waktu Eksekusi (s)	38.73	46.53	12.69
Beban Penggunaan CPU (%)	39.33	63.00	86.67
Memory Usage (%)	6.2	6.4	9.43

Hasil pengujian *multi node* pada menunjukkan bahwa GlusterFS melakukan kinerja lebih cepat dan lebih stabil dibandingkan HDFS. Pada operasional *write file* GlusterFS menunjukkan rata-rata *throughput* sebesar 45,04 MBps, waktu eksekusi selama 56,68 detik, beban penggunaan CPU sebesar 50% dan penggunaan memori sebesar 3,3%. Sedangkan pada HDFS menunjukkan rata-rata *throughput* sebesar 23,9 MBps, waktu eksekusi selama 90,88 detik, beban penggunaan CPU sebesar 74,33% dan penggunaan memori sebesar 5,27%.

Pada operasional *read file* GlusterFS menunjukkan rata-rata *throughput* sebesar 60 MBps, waktu eksekusi selama 42,63 detik, beban penggunaan CPU sebesar 51,16% dan penggunaan memori sebesar 6,3%. Sedangkan pada HDFS menunjukkan rata-rata *throughput* sebesar 222,15 MBps, waktu eksekusi selama 12,69 detik beban penggunaan CPU sebesar 86,67% dan penggunaan memori sebesar 9,43%. Hasil perbandingan kinerja *file* system mode multi *node* dapat dilihat pada Gambar 5.5 dan Gambar 5.6 berikut:



Gambar 5.5 : Grafik hasil perbandingan kenerja mode *multi node write file*



Gambar 5.6 : Grafik hasil perbandingan kenerja mode *multi node read file*

5.3 Pengujian Skenario Ketiga : Variasi Ukuran *Block*

Pengujian skenario ketiga adalah pengujian variasi ukuran *block. Block* merupakan unit atau bagian terkecil dari sebuah data dengan satuan *byte*. Ukuran maksimum sebuah *block* berbeda-beda setiap *file system.* Pada GlusterFS

ukuran maximum *block* adalah 32 MB, sedangkan pada HDFS adalah 128 MB. Pada pengujian skenario variasi ukuran *block,* disini akan digunakan ukuran *block* sebesar 1 MB, 2 MB, 4 MB, 8 Mb, 16 MB dan 32 MB. Tujuan dari pengujian ini adalah untuk melihat pengaruh perubahan ukuran *block file* terhadap kinerja *file system,* dengan parameter uji yang digunakan adalah *throughput.* Prosedur pengujian variasi ukuran *block* adalah sebagai berikut:

- Pastikan semua *node (node master, node slave)* telah aktif dan terhubung untuk setiap *file system*.
- Melakukan konfigurasi untuk merubah ukuran *block file* secara bertahap, mulai dari 1 MB, 2 MB, 4 MB, 8 MB, 16 MB dan terakhir 32 MB.
- Menjalankan perintah operasional penulisan dan pembacaan file dengan ukuran sebesar 2 GB.
- Hasil dari pengujian tersebut dimasukkan ke dalam tabel untuk selanjutnya dilakukan analisis perbandingan kinerja *file system.*

5.3.1 Hasil Pengujian dan Analisis Data

Untuk pengujian skenario variasi ukuran *block*, parameter pengujian yang digunakan hanyalah *throughput.* Hal ini dilakukan karena dalam GlusterFS tidak terdapat mekanisme pengaturan ukuran *block file* yang disimpan. Ukuran *block file* hanya bisa dilihat/dicek setelah menjalankan operasional *write/read file* dengan menjalankan perintah:

```
# gluster volume top [nama_volume] [write-perf|read-perf] bs
[ukuran_blocksize_dalam_byte] count [jumlah] brick
[node/:nama_brick] list-cnt [jml_list_count]
```

Hasil dari eksekusi perintah tersebut seperti terlihat pada

```
root@Gluster2:/home/gluster2# gluster volume top vol-Replikasi write-perf bs 1048576
count 1 brick gfs1:/replikasi1/ list-cnt 5
Brick: gfs1:/replikasi1
Throughput 1099.14 MBps time 0.0010 secs
MBps Filename                                   Time
==== ========                                   ====
   0 /report.xls                                2018-07-11 06:45:21.495841
   0 /rep1                                      2018-07-11 06:45:16.280871
   0 /report.xls                                2018-07-11 03:06:40.842139
   0 /report.xls                                2018-07-11 03:04:07.074833
   0 /report.xls                                2018-07-11 03:00:57.886485
```

Gambar 5.7 : *Gluster volume top*

Pada *file system* HDFS, untuk mengubah ukuran *block file*-nya tinggal merubah isi dari file konfigurasi hdfs-site.xml. Pada variabel dfs.block.size, ubah nilai *(value) block size* dalam ukuran *byte*. Jika pengisian nilai *block size* tidak sesuai dengan ukuran *bytes*, maka operasional *write/read file* tidak bisa dijalankan. Berikut adalah contoh pengisian nilai *(value)* ukuran *block file* sebesar 1 MB:

```
<property>
 <name>dfs.block.size</name>
 <value>1048576</value
</property>
```

Hasil dari pengujian skenario variasi ukuran *block* dapat dilihat pada Tabel 5.8 untuk operaional *write file* dan Tabel 5.9 untuk operasional *read file.*

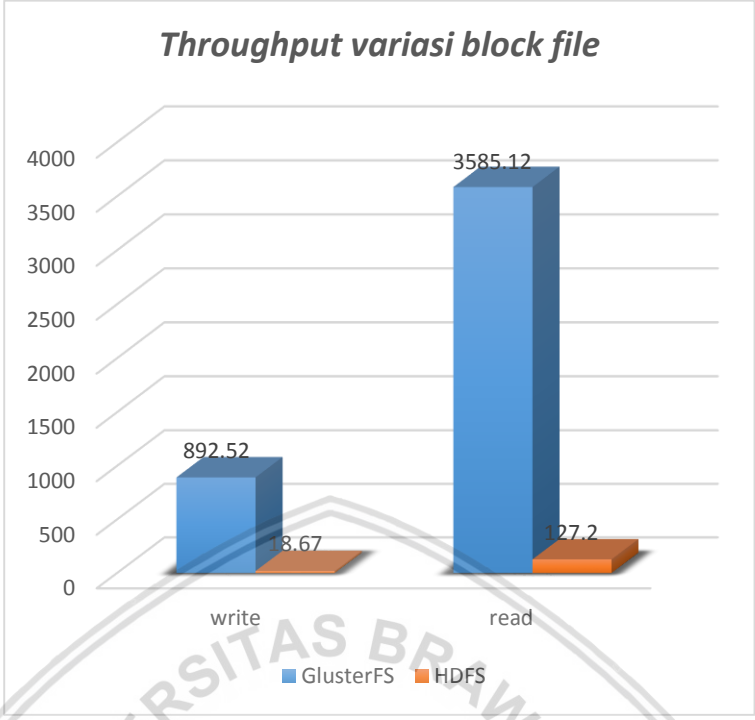
Tabel 5.8 : Hasil perbandingan nilai *throughput (MBps)* variasi ukuran *block write file*

Ukuran Block	GlusterFS-*Replicate*	GlusterFS-*Stripe*	HDFS
1 MB	1099.14	1685.81	13.29333
2 MB	1131.15	1488.4	15.35
4 MB	1354.31	1321.46	18.50333
8 MB	815.93	512.34	19.97667
16 MB	539.36	297.02	21.64667
32 MB	229.73	235.55	23.27333
Rata-rata	**861.60**	**923.43**	**18.67**

Tabel 5.9 : Hasil perbandingan nilai *throughput (MBps)* variasi ukuran *block read file*

Ukuran Block	GlusterFS-*Replicate*	GlusterFS-*Stripe*	HDFS
1 MB	3542.49	3840.94	84.20333
2 MB	3477.86	3666.35	87.54667
4 MB	4275.54	3276.8	161.0833
8 MB	3947.58	3618.9	132.0933
16 MB	2984.74	3604.13	138.8667
32 MB	3080.08	3706.03	159.38
Rata-rata	**3551.38**	**3618.86**	**127.20**

Mekanisme yang berbeda untuk pengujian variasi *block file* memberikan hasil dengan perbedaan yang besar antara GlusterFS dan HDFS. Operasional *write file* pada GlusterFS menunjukkan rata-rata *throughput* sebesar 892.52 MBps, sedangkan pada HDFS menunjukkan rata-rata *throughput* sebesar 18,67 MBps. Operasional *read file* pada GlusterFS menunjukkan rata-rata *throughput* sebesar 3585,12 MBps, sedangkan pada HDFS menunjukkan rata-rata *throughput* sebesar 127,2 MBps.



Gambar 5.8 : Perbandingan rata-rata nilai *throughput* variasi ukuran *block*

5.4 Pengujian Skenario Keempat : Variasi Jumlah *Brick* (*Replication Factor)*

Pengujian skenario yang keempat adalah variasi jumlah *brick. Brick* adalah direktori pada *node* yang berperan sebagai tempat tersimpannya file dan terhubung dengan setiap *node* pada *file system* melalui *mount point.* Jumlah replikasi menentukan di mana saja nantinya file tersimpan. Dalam pengujian ini jumlah replikasi file yang ditentukan adalah mulai dari 2 replikasi sampai 5 replikasi dengan kondisi *multi node*, yakni di mana *node master* terhubung dengan 2 *node slave* di masing-masing *file system*. Skenario pengujian ini dimaksudkan untuk mengetahui pengaruh jumlah replikasi file terhadap kinerja *file system* dengan parameter uji adalah *throughput* dan beban penggunaan CPU. Prosedur pengujian skenario keempat adalah sebagai berikut:

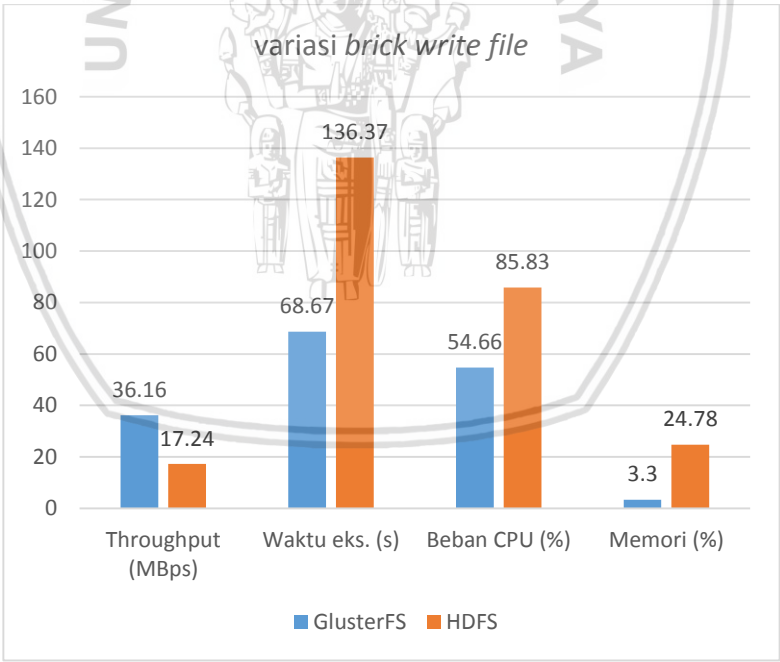
- Pastikan semua *node (node master, node slave)* telah aktif dan terhubung untuk setiap *file system*.
- Melakukan konfigurasi dengan menetukan jumlah replikasi file pada masing-masing *file system* mulai dari 2 replikasi hingga 5 replikasi.
- Menjalankan perintah operasional penulisan dan pembacaan file dengan ukuran sebesar 2 GB dan diulang sebanyak 3 kali setiap replikasi file.
- Hasil dari pengujian tersebut dimasukkan ke dalam tabel untuk selanjutnya dilakukan analisis perbandingan kinerja *file system.*

5.4.1 Hasil Pengujian dan Analisis Data

Hasil pengujian skenario variasi jumlah *brick* (pada GlusterFS) atau faktor replikasi (pada HDFS) menunjukkan bahwa pada saat operasional *write file* GlusterFS memiliki kinerja yang lebih cepat dan ringan dibanding HDFS. Pada Tabel 5.10, terlihat bahwa GlusterFS memperoleh nilai rata-rata *throughput* sebesar 36.16 MBps, waktu eksekusi selama 68,67 detik, beban penggunaan CPU sebesar 54,66% dan penggunaan memori sebesar 3,3%. Pada HDFS diperoleh nilai rata-rata *throughput* sebesar 17,24 MBps, waktu eksekusi selama 136,37 detik, beban penggunaan CPU sebesar 85,83% dan penggunaan memori sebesar 24,78%.

Tabel 5.10 : Hasil perbandingan variasi jumlah *brick/replication factor write file*

Parameter Uji	GlusterFS-*Replicate*	GlusterFS-*Stripe*	HDFS
Throughput (Mbps)	24.13	48.20	17.24
Waktu Eksekusi (s)	91.93	45.41	136.37
Beban Penggunaan CPU (%)	64.08	45.25	85.83
Memory Usage (%)	3.37	3.30	24.78



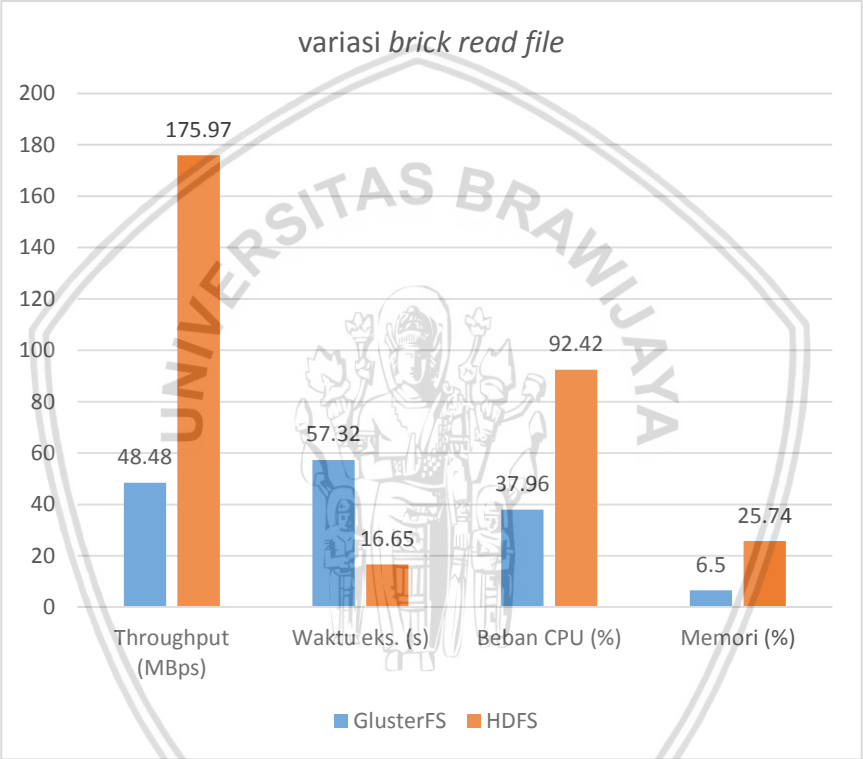
Gambar 5.9 : Grafik hasil perbandingan pengujian variasi *brick write file*

Pada saat operasional *read file* GlusterFS memiliki kinerja yang lebih lambat dan ringan dibanding HDFS. Pada Tabel 5.11, terlihat bahwa GlusterFS memperoleh nilai rata-rata *throughput* sebesar 48,48 MBps, waktu eksekusi selama 57,32 detik, beban penggunaan CPU sebesar 37,96% dan penggunaan memori sebesar 6,5%. Pada HDFS diperoleh nilai rata-rata *throughput* sebesar 175,97 MBps, waktu

eksekusi selama 16,65 detik, beban penggunaan CPU sebesar 92,42% dan penggunaan memori sebesar 25,74%.

Tabel 5.11 : Hasil perbandingan variasi *brick/replication factor read file*

Parameter Uji	GlusterFS-*Replicate*	GlusterFS-*Stripe*	HDFS
Throughput (Mbps)	67.63	29.34	175.97
Waktu Eksekusi (s)	33.81	80.84	16.65
Beban Penggunaan CPU (%)	60.17	35.75	92.42
Memory Usage (%)	6.61	6.46	25.74



Gambar 5.10 : Grafik hasil perbandingan pengujian variasi *brick read file*

5.5 Pengujian Skenario Kelima : Variasi Rekayasa Kesalahan Sistem

Pengujian skenario yang kelima adalah variasi rekaya kesalahan yang terjadi pada *file system* ketikan melakukan proses *write file*. Ketika proses *write file* sedang berlangsung beberapa *node* akan sengaja dimatikan, dimulai dengan mematikan 1 *node slave*, kemudian 2 *node slave* dan yang terakhir adalah mematikan *master node.* Tujuan dilakukan pengujian skenario ini adalah untuk melihat apakah file yang ditulis ke dalam *node* itu berhasil ditulis seluruhnya atau tidak. Prosedur pengujian skenario ini adalah sebagai berikut:

- Pastikan semua *node (node master, node slave)* telah aktif dan terhubung untuk setiap *file system*.
- Menjalankan perintah operasional penulisan dan pembacaan file dengan ukuran sebesar 2 GB.
- Ketika operasional penulisan file berlangsung, 1 *node slave* dimatikan.
- Menghubungkan kembali semua node dan menjalankan operasional penulisan file berukuran 2 GB.
- Ketika operasional penulisan file berlangsung, 2 *node slave* dimatikan.
- Menghubungkan kembali semua node dan menjalankan operasional penulisan file berukuran 2 GB.
- Ketika operasional penulisan file berlangsung, *node master* dimatikan.
- Hasil dari pengujian tersebut dimasukkan ke dalam tabel untuk selanjutnya dilakukan analisis perbandingan kinerja *file system.*

5.5.1 Hasil dan Analisis Data

Tabel 5.12 : Ouput file pada setiap *node* setelah 1 *node slave* dimatikan

Kondisi	Data/File			Besar Ukuran (MB)		
	Master	Slave1	Slave2	Master	Slave1	Slave2
HDFS	Ada	Ada	Ada	2048	1720	387
GlusterFS (stripe)	Null	Ada	Ada	0	1495	1493
GlusterFS (replika)	Null	Ada	Ada	0	2147	1398

Tabel 5.13 : Ouput file pada setiap *node* setelah 2 *node slave* dimatikan

Kondisi	Data/File			Besar Ukuran (MB)		
	Master	Slave1	Slave2	Master	Slave1	Slave2
HDFS	Ada	Ada	Ada	2048	306	306
GlusterFS (stripe)	Null	Ada	Ada	0	743	744
GlusterFS (replika)	Null	Ada	Ada	0	1650	1545

Tabel 5.14 : Ouput file pada setiap *node* setelah *node master* dimatikan

Kondisi	Data/File			Besar Ukuran (MB)		
	Master	Slave1	Slave2	Master	Slave1	Slave2
HDFS	Ada	Ada	Ada	780	957	951
GlusterFS (stripe)	Null	Ada	Ada	0	371	371
GlusterFS (replika)	Null	Ada	Ada	0	1837	1837

Berdasarkan Tabel 5.12, Tabel 5.13 dan Tabel 5.14, HDFS lebih bagus dalam hal penangan kesalahan sistem daripada GlusterFS. Meskipun 1 atau 2 *node slave*-nya tidak aktif atau tiba-tiba saja mati, *node master* masih menjalankan operasional *write/read file.* Tidak seperti GlusterFS, yang apabila salah satu *node slave*-nya tidak aktif atau tiba-tiba saja mati, maka operasional *write/read* file akan langsung dihentikan dan sehingga apabila semua *node* kembali terhubung maka file dengan ukuran sedemikianlah yang akan didistribusikan ke setiap *node.*

BAB 6 PENUTUP

6.1 Kesimpulan

Ditinjau dari hasil implementasi, keseluruhan pengujian skenario dan analisis perbandingan kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *stripe* dan *replicate,* dapat disimpulkan bahwa:

1. Pembangunan lingkungan pengujian untuk membandingkan kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *striped* dan *replicate* dilakukan dengan membuat modul sistem file terdistribusi secara virtual mengguanakan VirtualBox. Dengan dukungan *tool* IOzone pada GlusterFS dan DFSIO pada HDFS yang digunakan dalam operasional *write/read file*.
2. Parameter yang digunakan dalam perbandingan pengukuran kinerja *file system* GlusterFS dan HDFS dengan skenario distribusi *stripe* dan *replicate* adalah *throughput,* waktu eksekusi, beban penggunaan CPU dan *memori usage.*
3. Skenario pengujian yang digunakan dalam membandingkan *file system* GlusterFS dan HDFS adalah dengan menerapkan 5 skenario pengujian yang berbeda, yakni variasi ukuran file, variasi ukuran *block file*, variasi jumlah brick / *replication factor*, variasi jumlah *node* dan variasi pengangan *fault system*.
4. GlusterFS pada operasional *write file* diperoleh hasil rata-rata *throughput* sebesar 44,54 MBps, waktu eksekusi selama 58,54 detik, beban penggunaan CPU sebesar 54,83% dan penggunaan memori sebesar 3,6%. Sedangkan pada operasional *write file* diperoleh hasil rata-rata *throughput* sebesar 50.26 MBps, waktu eksekusi selama 52,76 detik, beban penggunaan CPU sebesar 52,34% dan penggunaan memori sebesar 6,4%.
5. HDFS pada operasional *write file* diperoleh hasil rata-rata *throughput* sebesar 24,1 MBps, waktu eksekusi selama 104,94 detik, beban penggunaan CPU sebesar 66,79% dan penggunaan memori sebesar 18,4%. Sedangkan pada operasional *write file* diperoleh hasil rata-rata *throughput* sebesar 194,37 MBps, waktu eksekusi selama 16,01 detik, beban penggunaan CPU sebesar 86,9% dan penggunaan memori sebesar 18,5%.
6. Dari hasil pengujian skenario penanganan kesalahan, diperoleh hasil bahwa HDFS lebih bagus dalam menangani kesalahan sistem. Operasional penyimpanan tetap dijalankan dan data tetap tersimpan di semua *node*.

6.2 Saran

Saran yang dapat disampaikan untuk pengembangan dan penelitian lebih lanjut adalah sebagai berikut:

1. Perlu dilakukannya penelitian lebih lanjut, dengan menambah spesifikasi perangkat yang digunakan. Semakin bagus spefifikasinya tentu bisa mendapatkan hasil yang lebih baik.
2. Perlu dilakukan pengujian dengan varian format data, untuk mengetahui hasil yang lebih signifikan.

DAFTAR PUSTAKA

Ada-Europe, *Ada Reference Manual : Execution Time.* [online] Tersedia di: <http://www.adaic.org/resources/add_content/standards/05rm/html/RM-D-14.html> [Diakses pada 12 Desember 2017]

Donvito, G., Marzulli, G., dan Diacono, D., 2013. *Testing of Several Distributed File-System (HDFS, Ceph and GlusterFS) for Supprting The HEP Experiments Analisys.* Journal of Physics: Conference Series 513 (2014). IOP Publishing

Ehliar, A., dan Liu, D., 2004. *Benchmarking Network Processor*. Swedia: Dept. of Electrical Engineering Link¨oping University S-581 83 Link¨oping.

Ehrhardt, C., 2010. *CPU Time Accounting.* s.I.IBM.

Gluster Docs., *GlusterFS Documentation.* [online] Tersedia di: <https://docs.gluster.org /en/latest/Administrator-Guide/Setting-Up-*Volumes*/Setting-up-GlusterFS-*Volumes*.html> [Diakses pada 24 Februari 2017]

Gluster, 2011. *Cloud Storage for the Modern Data Center - An Introduction to Gluster Architecture.Ver:3.1.x.* Gluster Inc.

Gudanglinux, 2012. *Membangun Server NFS-Like Standalone Storage Menggunakan GlusterFS 3.2 di Debian Squeeze*, [online] Tersedia di: <http://gudanglinux.info/info2012/info/index.php/howtolinks/206-storage/1288-creating-an-nfs-like-standalone-storage-server-with-glusterfs-30x-on-debian-squeeze.html> [Diakses pada 12 Februari 2017]

Ivan, 2016. *Exploring Latest Tech Trends.* [online] Tersedia di: <https://mcpaw.com/how-to/what-is-heavy-memory-usage> [Diakses pada 28 Mei 2017]

Khusumanegara, P., 2014. *Analisis Performa Kecepatan MapReduce pada Hadoop Menggunakan TCP Packet Flow Analisys*. Depok: UI

Lam, C., 2011. *Hadoop in Action*. Greenwich: Manning Publications Co.

Lynch, D., 2017. *Running DFSIO MapReduce benchmark test.* [online] Tersedia di: <https://discuss.pivotal.io/hc/en-us/articles/200864057-Running-DFSIO-MapReduce-benchmark-test> [Diakses pada 21 Oktober 2017]

Maryanto, B., 2017. *Big Data dan Pemanfaatannya dalam Berbagai Sektor*. Media Informatika Vo.16 No.2. Bandung: STKI LIKMI.

Megantara, F., dan Warnars, H.L.H.S., 2016. *Implementasi Big Data untuk Pencarian Pattern Data Gudang pada PT. Mandiri (Persero) Tbk.* Jurnal Sisfotek Global. Tangerang: STMIK Bina Sarana Global.

Narendra, A.P., 2017. *Data Besar, Data Analisis, dan Pengembangan Kompetensi Pustakawan.* Salatiga: Fakultas Teknologi Informasi UKSW.

Scribd, 2013. Makalah: *File Service pada Sistem Terdistribusi*. [online] Tersedia di: <https://www.scribd.com/doc/138427963/Makalah-File-Service-pada-Sistem-Terdistribusi> [Diakses pada 24 Februari 2017]

Sekarwati, K.A., 2005. *File Services.* Staffsite Universitas Gunadarma. Tersedia di: <http://revida.staff.gunadarma.ac.id/Downloads/files/48802/07-File-Service.pdf> [Diakses pada 24 Februari 2017]

Sulistyo, W., dan Prasetyo, S.Y.J., 2014. *Analisis Kinerja Layanan Penyimpanan File Terdistribusi Cluster High-Availability Storage Dengan Menggunakan GlusterFS.* [pdf] Universitas Kristen Satya Wacana. Tersedia di: <http://repository.uksw.edu/bitstream/123456789/8766/3/T1_672011709_Full-text.pdf> [Diakses pada 24 Februari 2017]

Wahyudi, G., dan Hanggara, T., 2013. *Analisis Perbandingan Kinerja Antara Network File System (NFS) dan Primary Domain Controller (PDC) SAMBA*. Jurnal Ilmu Komputer (Vol.6 No.1): Universitas Udayana

Wikipedia, 2015. *Benchmark Computing*. [online] Tersedia di: <https://en.wikipedia.org/wiki/Benchmark_computing> [Diakses pada 10 Agustus 2017]

Yulianto, B., 2015. *Big Data - HDFS: Berawal dari Google untuk Big Data.* [online] Tersedia di: <https:/buangyulianto.blogspot.co.id/big-data-hdfs-berawal-dari-google-untuk-big-data.html> [Diakses pada 2 Februari 2017]

Zulfikar, A., 2015. *Konsep dan Implementasi GlusterFS*. [online] Tersedia di: <https://arifzulfikarp.blogspot.co.id/2015/06/konsep-dan-implementasi-glusterfs.html> [Diakses pada 12 Februari 2017]